Digital Beam Synthesis (DBS) for a High Capability Opto-Electronic Radar (HICAPOR)

JASON
MITRE

Digital Beam Synthesis (DBS) for a High Capability Opto-Electronic Radar (HICAPOR)

Study Leaders:
Alvin Despain
John Vesecky

JSR-97-230

September 1997

JASON
The MITRE Corporation
1820 Dolley Madison Boulevard
McLean, Virginia 22102-3481
(703) 883-6997

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
September 25, 1997

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
Digital Beam Synthesis (DBS) For A High Capability Opto-Electronic Radar (HICAPOR)

5. FUNDING NUMBERS
13-988534-04

6. AUTHOR(S)
A. Despain, J. Vesecky

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The MITRE Corporation
JASON Program Office
1820 Dolley Madison Blvd
McLean, Virginia 22102

8. PERFORMING ORGANIZATION REPORT NUMBER
JSR-97-230

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Code 111
800 North Quincy Street
Arlington, Virginia 22217

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
JSR-97-230

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Digital beam synthesis using a high capability opto-electronic radar (HICAPOR) by Alvin M. Despain and John F. Vesecky. This JASON study investigates the capabilities of HICAPOR by calculating the antenna beam patterns formed by typical implementations of this concept. A wide variety of para-meter choices are investigated and antenna patterns for HICAPOR are compared with conventional phased array and true time delay techniques of beam formation. The presentation begins with an introduction of the HICAPOR concept and how it fits within a high capability, very wide band radar. This is followed by presentation of a computer simulation of the beam and pulse forming capabilities of HICAPOR over a range of radar parameters. HICAPOR beamforming performance is then compared with the performance of other radar types. Finally conclusions are drawn and recommendations made.

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Outline

Digital beam synthesis using a high capability opto-electronic radar (HICAPOR) by Alvin M. Despain and John F. Vesecky.

This JASON study investigates the capabilities of HICAPOR by calculating the antenna beam patterns formed by typical implementations of this concept. A wide variety of parameter choices are investigated and antenna patterns for HICAPOR are compared with conventional phased array and true time delay techniques of beam formation. The presentation begins with an introduction of the HICAPOR concept and how it fits within a high capability, very wide band radar. This is followed by presentation of a computer simulation of the beam and pulse forming capabilities of HICAPOR over a range of radar parameters. HICAPOR beamforming performance is then compared with the performance of other radar types. Finally conclusions are drawn and recommendations made.

This page left blank intentionally.

HICAPOR

Opto-Electronic Control

- Conventional Low-Performance Phased Array Radar
  - Simple control of a single transmit beam
  - Moderate bandwidth, complex waveforms available
  - Large changes in frequency cause squint of beam for angles off boresight
  - Single receive beam only
  - No null steering capability
- High-Performance, True Time Delay (TTD) Steering Phased Array Radar
  - Use of optical delay devices allows large changes in frequency with no beam squint
  - Use of optical fibers makes signal transmission easier and more immune to interference
- High-Capability Opto-Electronic Radar (HICAPOR)
  - Combines optical signal transmission with direct digital synthesis of transmit signal at array element

Opto-Electronic Control

Conventional low-performance phased array radar suffers from a variety of short comings. In particular phase shift is used to control beam angle and this leads to degradation of the beam shape when frequency is shifted a large amount from the nominal operating frequency. This problem is called beam squint. One way to correct the squint problem for very large bandwidths or large changes in operating frequency is to use time delay to steer the antenna beam rather than phase shift. Time delay beam steering can be implemented electronically, but the size of waveguides and other electronic devices makes the implementation awkward. Optical fibers are very effective in the transmission of very wide bandwidth and time delay devices, such as the bifodal (a switched, binary tree of fiber optic delay lines), can be used to steer phased array antenna beams. However, there are drawbacks to optical delay lines, including stability, sensitivity to severe environments and production of long delays. Further, improvement can be made by using optical fibers to transmit microwave signals around the radar, but doing the time delay (along with waveform generation) by direct digital synthesis at each array transmit element.

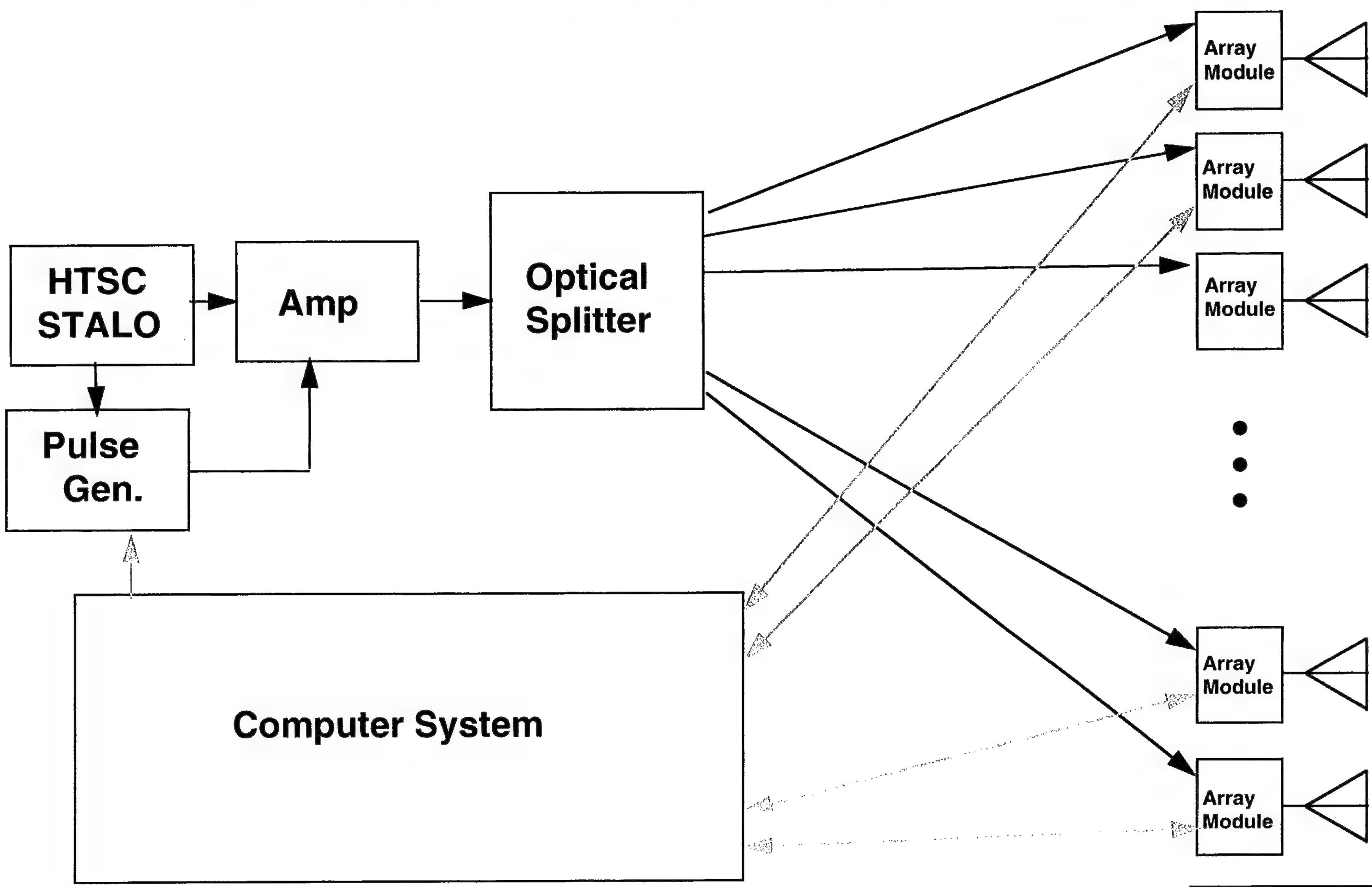
The HICAPOR Strawman

- ‘N’ Transmit Beams
  - Transmit waveform synthesis at each array element
- ‘N’ Receive Beams
  - A/D conversion at each antenna element
- Optical Fiber used for:
  - RF-Clock distribution
  - Synchronization
  - A/D data link to computer
  - Computer link to waveform generator
- Employ D/A waveform generator for both pulse forming and beam forming
- Optical Fiber/Devices are NOT used for beamforming!

The HICAPOR Strawman

To illustrate the HICAPOR concept we construct a strawman design and then compute the beam formation capability of HICAPOR for a variety of parameters. We also compare the HICAPOR capability with a variety of other options including conventional phased array radar and true time delay beam steering. In the HICAPOR strawman design optical fiber is used extensively for RF clock distribution, synchronization, A/D data link to computer and computer link to waveform generator. This is illustrated on the following viewgraphs. However, optical fiber/devices are NOT used for beamforming. Instead a electronic digital D/A is used for both waveform generation and beam formation. This means that the waveform timing must be at the 0.1 ns level to accomplish the beam forming function. The HICAPOR concept and how it fits into a high capability array antenna radar is illustrated in the following viewgraphs.

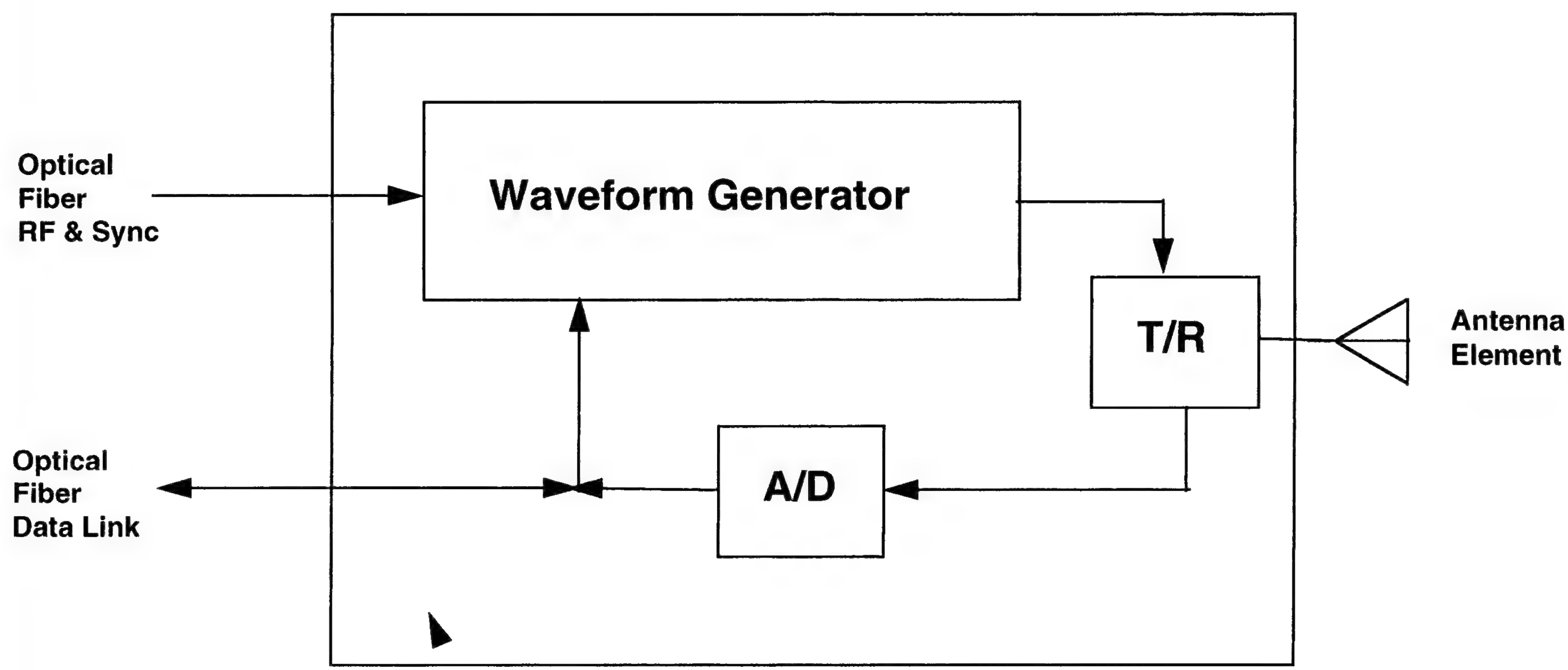
HICAPOR Architecture



HICAPOR Architecture

This viewgraph illustrates the radar concept of HICAPOR. Much of the radar is conventional, but advanced components are used in several places. To decrease phase noise for better Doppler discrimination of low cross section targets, a high-temperature superconducting stable local oscillator (HTSC STALO) is used. Communication of RF and digital signals around the radar is done by fiber optics. This includes the master RF carrier signal for radar transmissions, time synchronization signals as well as the digital information needed by the array modules for beam formation, waveform generation and other radar functions. The received radar echo signal is digitized at the array module level and this information is transmitted by optical fiber to the radar's computer system for receive beam formation, Doppler processing and other data analysis functions.

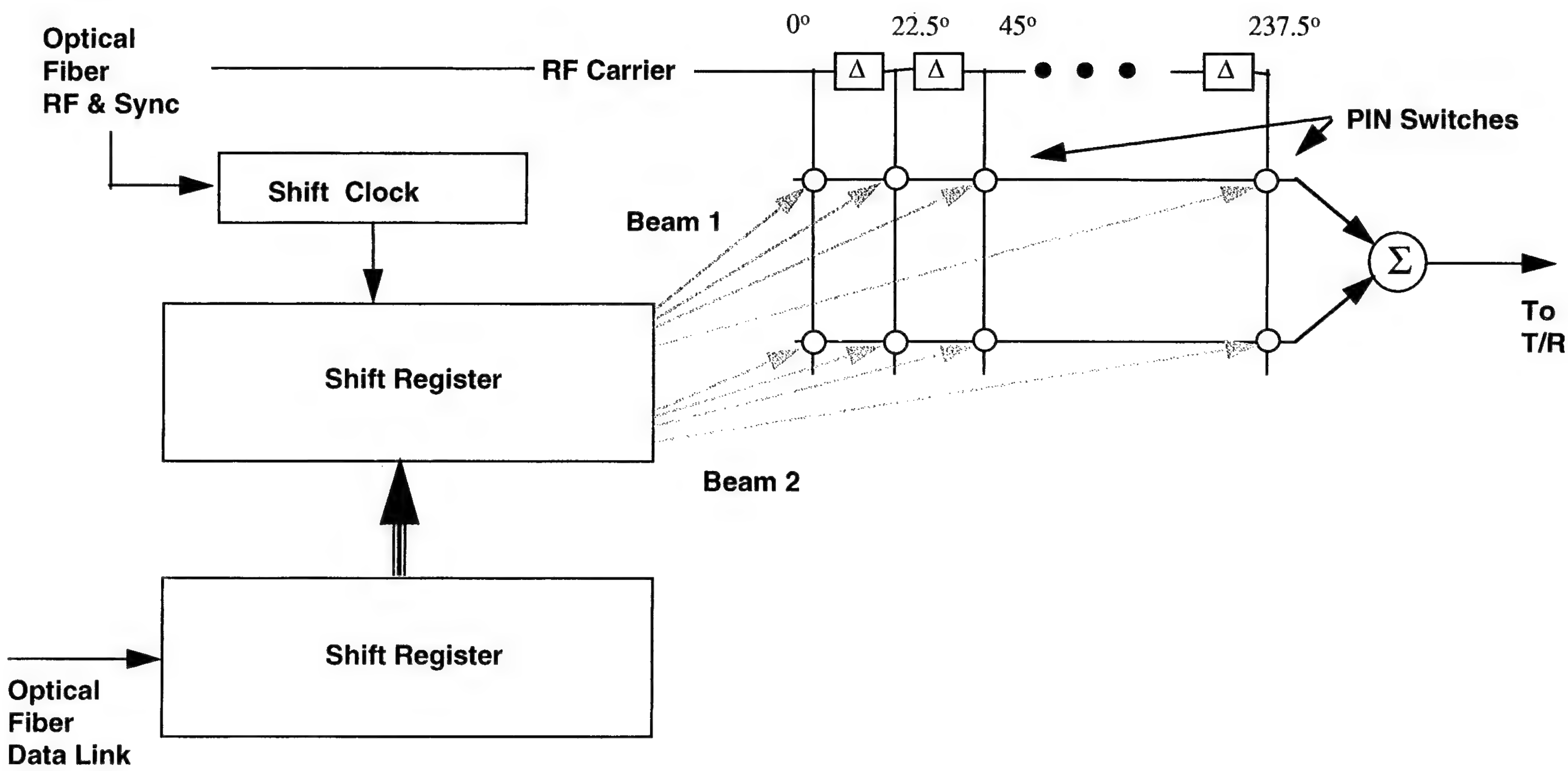
Array Module



Array Module

Each array module contains devices for generation of the radar transmit signal as well as the A/D conversion of the received echo signal. The waveform generator receives RF carrier, synchronization and waveform and beam forming instructions by optical fiber links to the radar computer system. The synchronization pulse begins each pulse forming process. This generator then manufactures the output signal to be transmitted by the particular array module at say 10 GHz. This generator is the heart of the HICAPOR concept and is described in more detail on the following viewgraphs. On receive, the received signal is down converted to baseband and then digitized with 15 bit accuracy at a typical bandwidth of say 10 MHz. Although the individual echo signals at each array module are digitized at only 15 bit accuracy, the aggregate accuracy of the combination of many array module signals in the beamforming process increases the accuracy significantly for a large array. For a large array this increased accuracy by combination can result in 20 significant bits and a dynamic range of say 120 dB in the Doppler-processed signal.

Waveform Generator

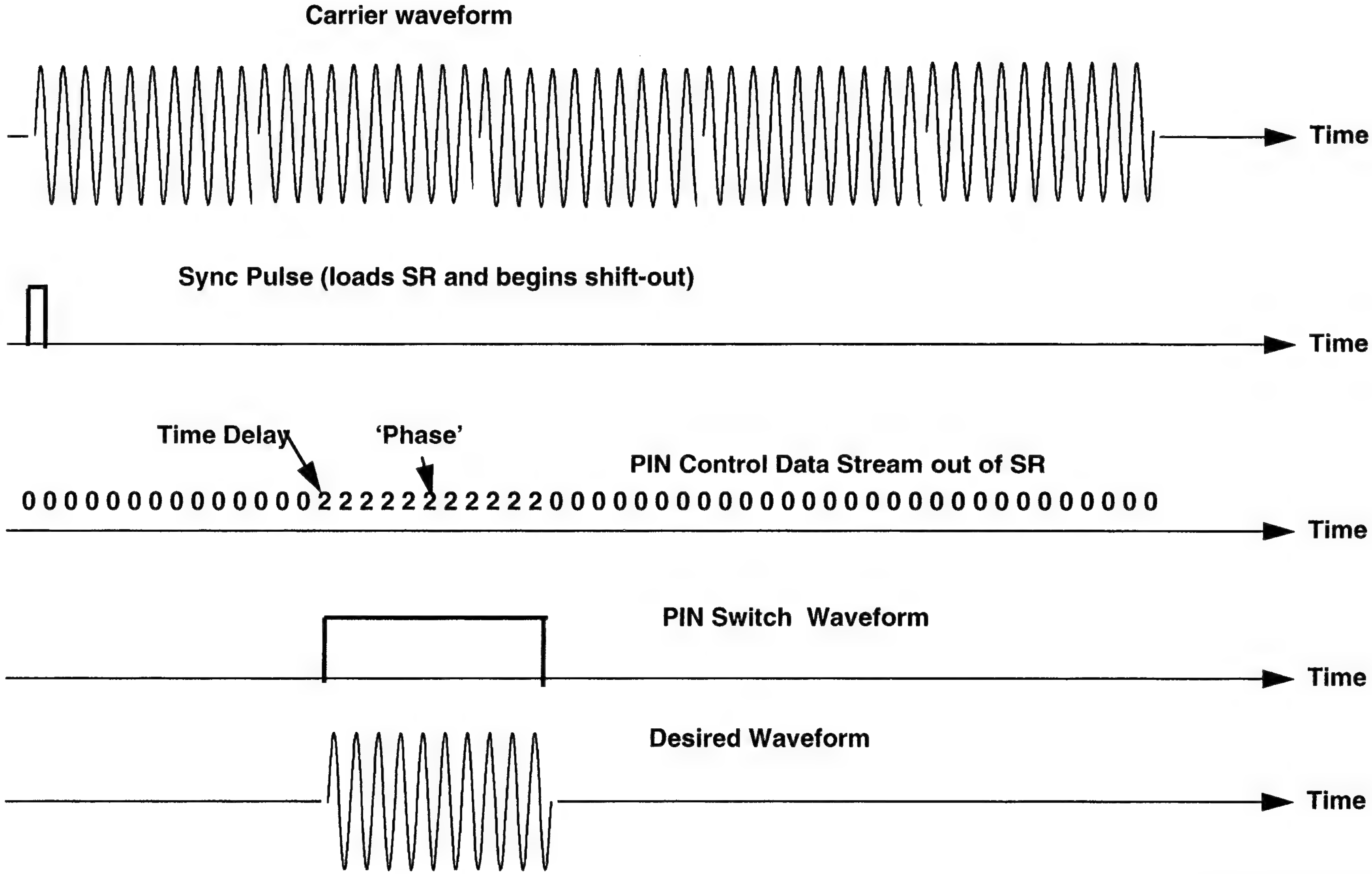


JASON HICAPOR

Waveform Generator

The waveform generator is controlled by a digital data stream providing the successive loads to a large shift register (bottom left). This data stream contains the information necessary to generate the output waveform, complete in both waveform characteristics and location in time so that the desired beam is formed. The upper shift register contains the information for one complete output pulse. Thus, at instant of time, i.e. each 0.1 ns interval, this upper shift register shifts out eight binary numbers to control the eight PIN diodes. Throughout the radar pulse this upper shift register keeps shifting out 8 bit sequences to control the PIN diodes. For a 100 ns pulse length this would amount to 1000 8-bit sequences. When the pulse is finished the lower shift register loads the upper register with the information for the next pulse. This can be done at leisure since there is typically about 1 ms between pulses. The fact that each 8-bit sequence controls the signal phase allows the beam to be steered with high efficiency. The effects of changing the size of the shift register and the number of PIN diode control points is explored in the viewgraphs that follow somewhat further on. In this implementation we show how multiple beam capability is introduced. Namely, a second row of PIN diodes uses the same RF carrier and phase shifts, but with different 8-bit sequences corresponding to another waveform in a different direction. Each additional beam requires a doubling of the size of the upper shift register.

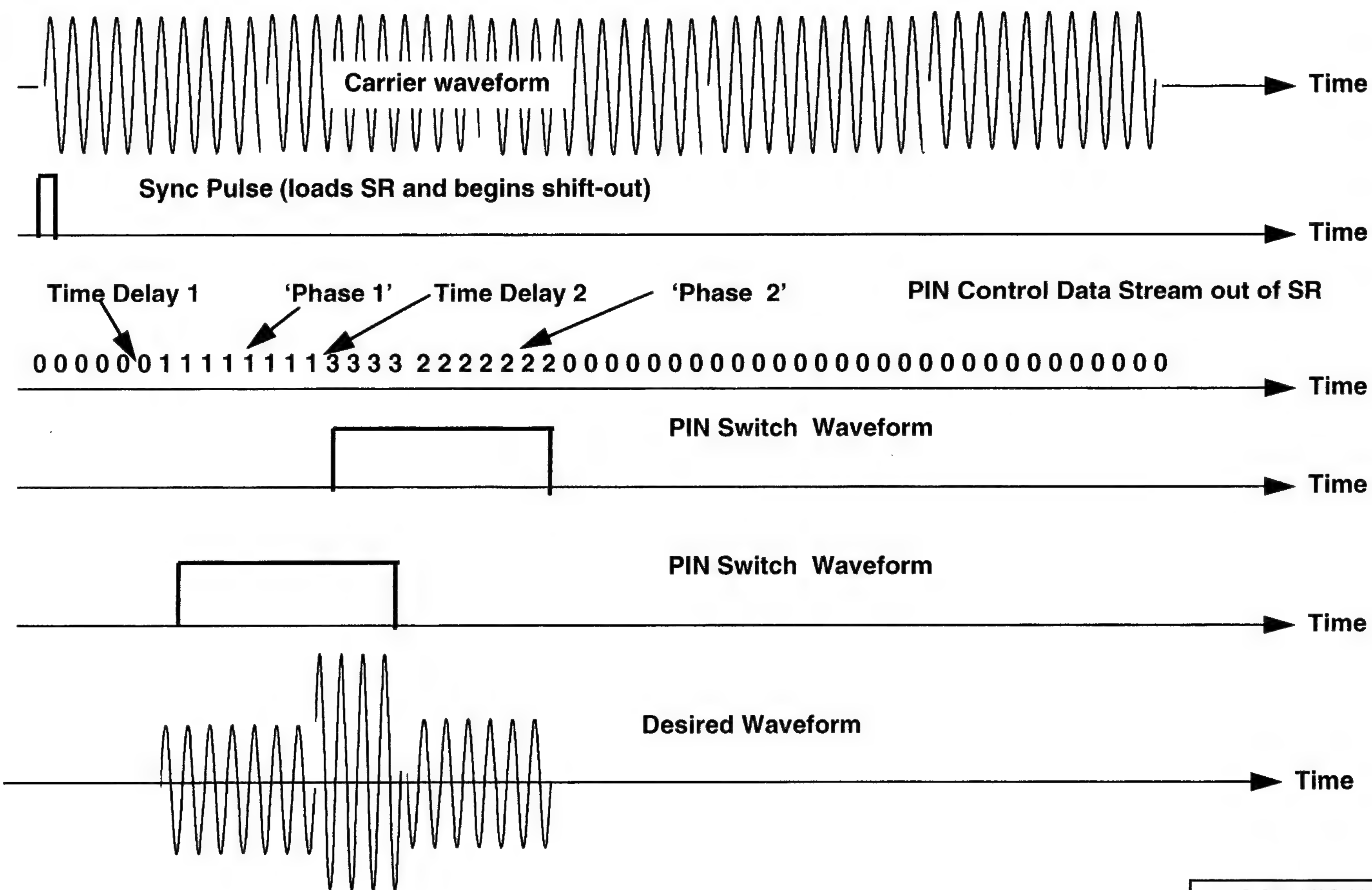
Simple Waveforms



Simple Waveforms

Here we illustrate the HICAPOR pulse formation scheme with a simple waveform consisting of a single pulse that lasts for eleven shift register clock periods. At the top is the carrier waveform that supplies the signal to the phase shifters. The next time series below shows the sync pulse that loads the upper shift register from the lower one and begins the shift out to form one complete pulse. The third time series from the top shows which PIN diode is turned on. In this case all the PIN diodes are turned off except number two which is turned on for eleven shift out periods. The fourth time series shows the switch waveform for diode number two. The output waveform is shown at the bottom of the viewgraph.

Waveforms-Two Beams



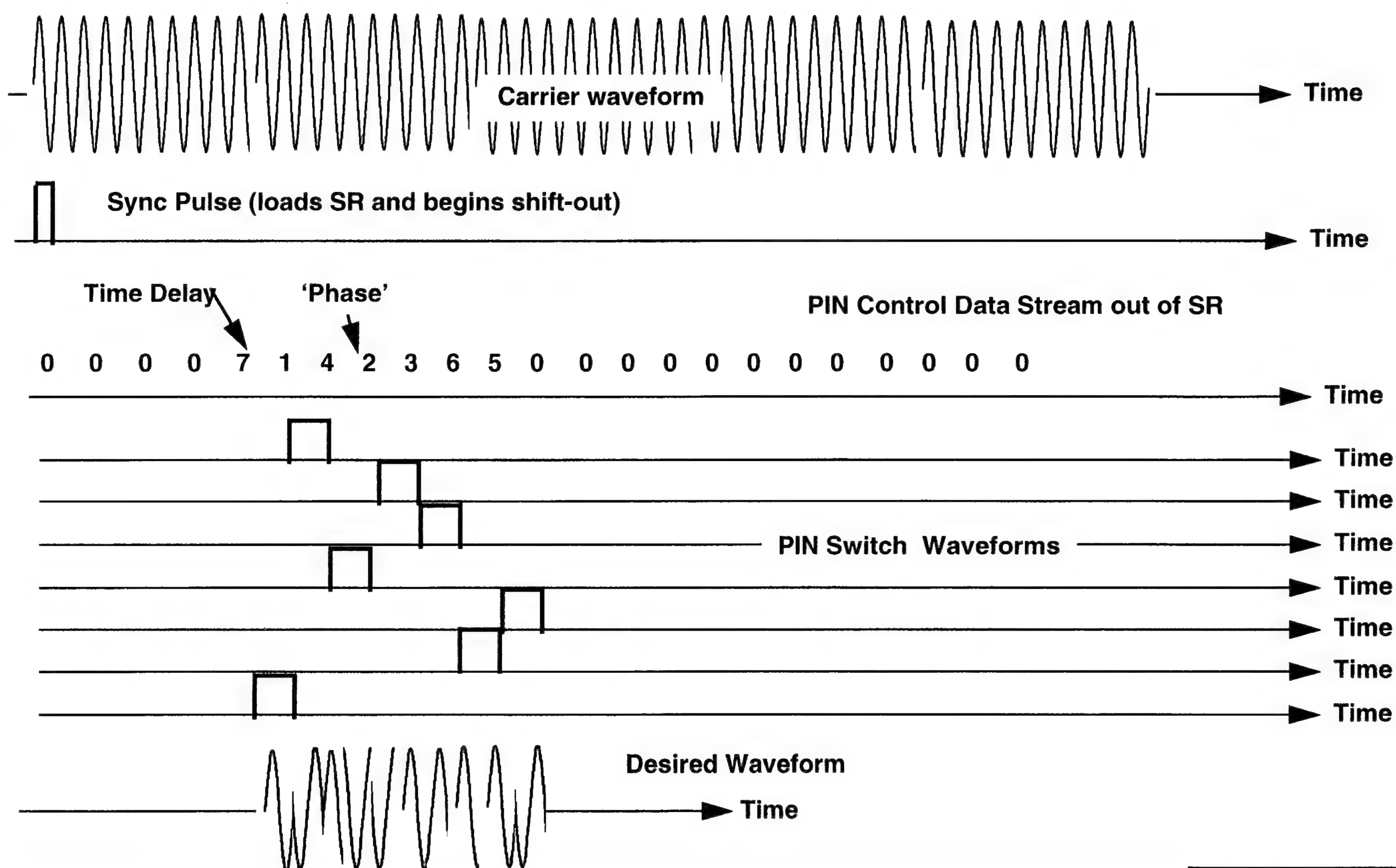
JASON HICAPOR

Waveforms-Two Beams

Here we illustrate the HICAPOR pulse formation scheme for two independent beams. At the top is the carrier waveform that supplies the signal to the phase shifters. The next time series below shows the sync pulse that loads the upper shift register from the lower one and begins the shift out to form one complete pulse. The third time series from the top shows which PIN diodes are turned on. At first only one pulse is specified to be on. Soon codes in the shift register begin to specify both pulses. Later the first pulse ends and finally the output is specified to be zero. The fourth time series shows the switch waveform for the pin diodes. The output waveform is shown at the bottom of the viewgraph.

Waveforms-Phase Encoding

[Pulse Compression]



Waveforms-Phase Encoding
[Pulse Compression]

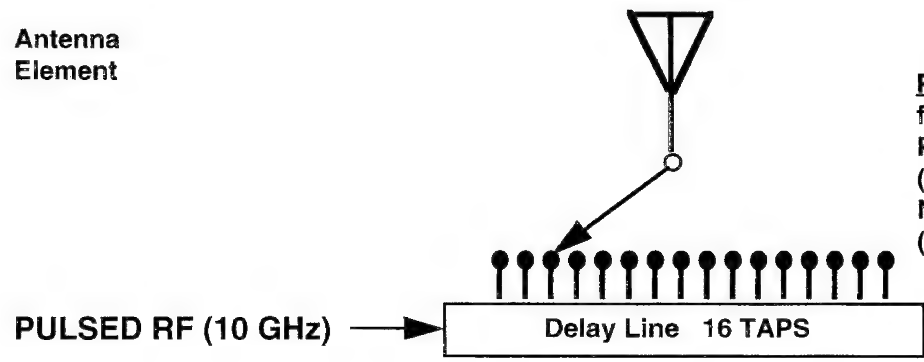
Here we illustrate the HICAPOR pulse formation scheme for pulse compression. At the top is the carrier waveform that supplies the signal to the phase shifters. The next time series below shows the sync pulse that loads the upper shift register from the lower one and begins the shift out to form one complete pulse. Different codes are used to specify which phase to transmit at each time. The next set of time series from the top shows which PIN diodes are turned on. The output wave-form is shown at the bottom of the viewgraph.

Examples for Simple Pulse Radars

Parameters
f = 10 GHz
Phase Res. = $\pi/8$
(16 Phases)
N = 100 Elements
(10,000 total)

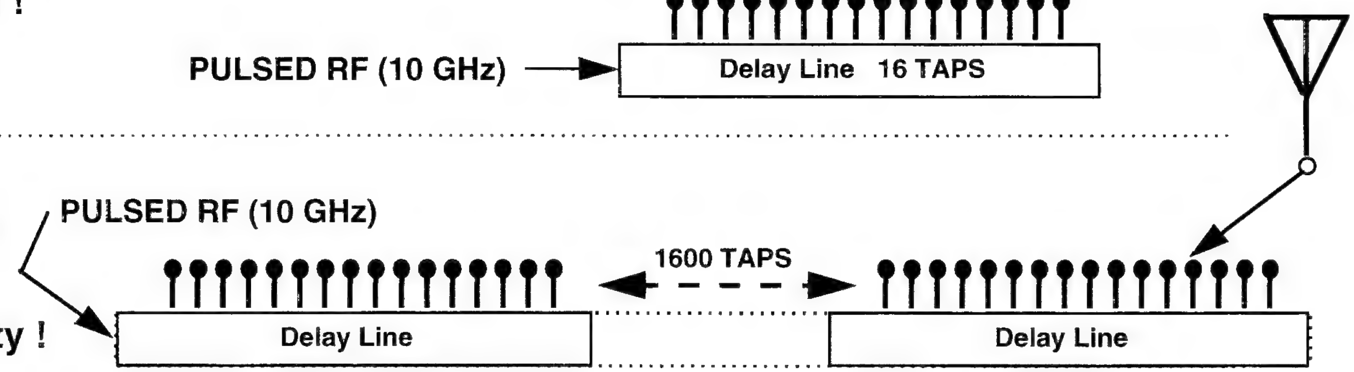
Classic Phased Array
16 taps of a delay line

Problem: Squinting !



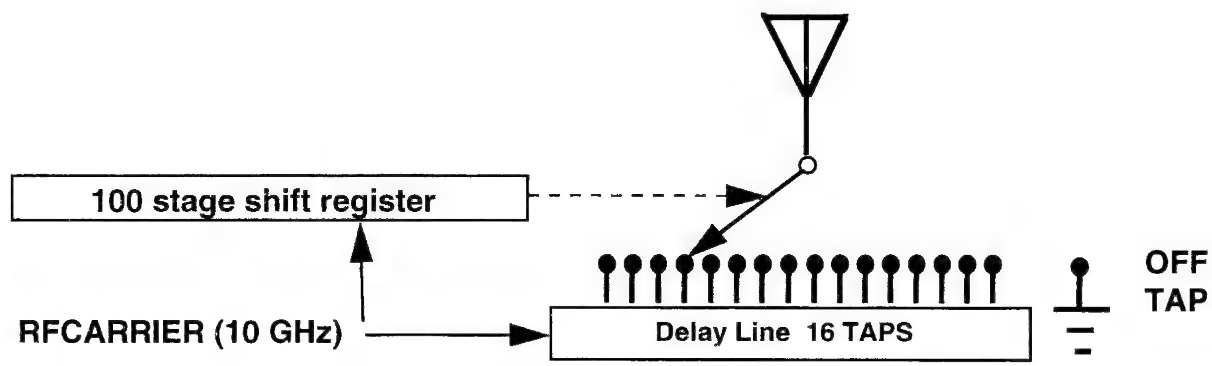
Time Delay Array
16 x 100 = 1600 taps
of a delay line

Problem: Complexity !
Many km of fiber



HICAPOR
16 taps &
100 stage shift register
10 GHz Clock

Simple & Solves Squint problem, adds flexibility



Examples for Simple Pulse Radar

This viewgraph illustrates the important differences between the classic phased array, the true time delay (TTD) array and the HICAPOR concept. In each case we consider an individual array module and show what is required to generate the transmitted signal for some desired antenna beam direction. The classic phased array at the top the antenna can be hooked to any of 16 taps on a delay line. The delay line taps are such that a phase change of 360° is distributed over the 16 taps. Thus, the delay line acts like a phase shifter, which is what we want for this comparison. The problem for the classic phase array antenna is the squinting problem, discussed above. In the time delay array case the delay line requires enough taps to allow each element to select the correct delay. For a 100 element array this requirement means that the tapped delay line at each element must have some 16 x 100 delays. Such a delay line can be implemented using fiber optic cables and optical switches. Schemes, such as the bifodal which switches in or out a number of optical fibers to construct the correct delay, can reduce the complexity. Still the problem is one of complexity as the optical delay lines are long if the change in time delay is done by changing optical wavelength or many optical

Examples for Simple Pulse Radar
(concluded)

switches are required if one changes delay by propagating through a different physical set of fibers. The HICAPOR concept uses a 100 stage shift register to digitally control the delay and to digitally control the 16 stage delay line as shown in the viewgraph. Thus, the HICAPOR system uses optical fibers where they can do the most good, i.e. in digital and analog signal communication within the radar. The digital waveform generator can be made very small and thus fit nicely within an array module, whereas current optical delay devices would make this fit much more difficult. We also think that the cost of HICAPOR modules would be significantly less than their optical counterparts.

SIMULATION OF THE DIGITAL BEAM SYNTHESIS SYSTEM

Digital Beam Synthesis (DBS)

- DBS is a component of the HICAPOR system that forms a RADAR transmitted beam.
- DBS is an invention to greatly reduces cost and size of a true-time delay beamformer.
- DBS separately modulates a carrier for each array antenna element at exactly the right time for that element.
- In DBS delay is provided by a digital circuit under computer control.
- A shift register is one example of such a digital control circuit.
- One form of DBS employs a vernier scheme such that the shift register provides a control character that selects a desired phase of the RF carrier.

Digital Beam Synthesis (DBS)

The DBS invention greatly reduces the cost and size of RADAR systems by moving the modulation process after the delay process. Thus the delays can be specified in a simple digital shift register or other digital circuit under computer control.

Digital Representation

- DBS, being digital, employs discrete step approximations for the radar signals.
- If the steps are too gross, error artifacts will appear in the beam pattern.
- The DBS and six other beam forming systems have been simulated.
- Artifacts unique only to DBS can be determined by comparison to the other systems.

Digital Representation

The DBS system(as well as other systems) employ discrete approximations for the RADAR signals. If these are too crude, then undesireable artifacts will appear in the transmitted RADAR beam. By simulating DBS and other systems , the source of the artifacts can be determined.

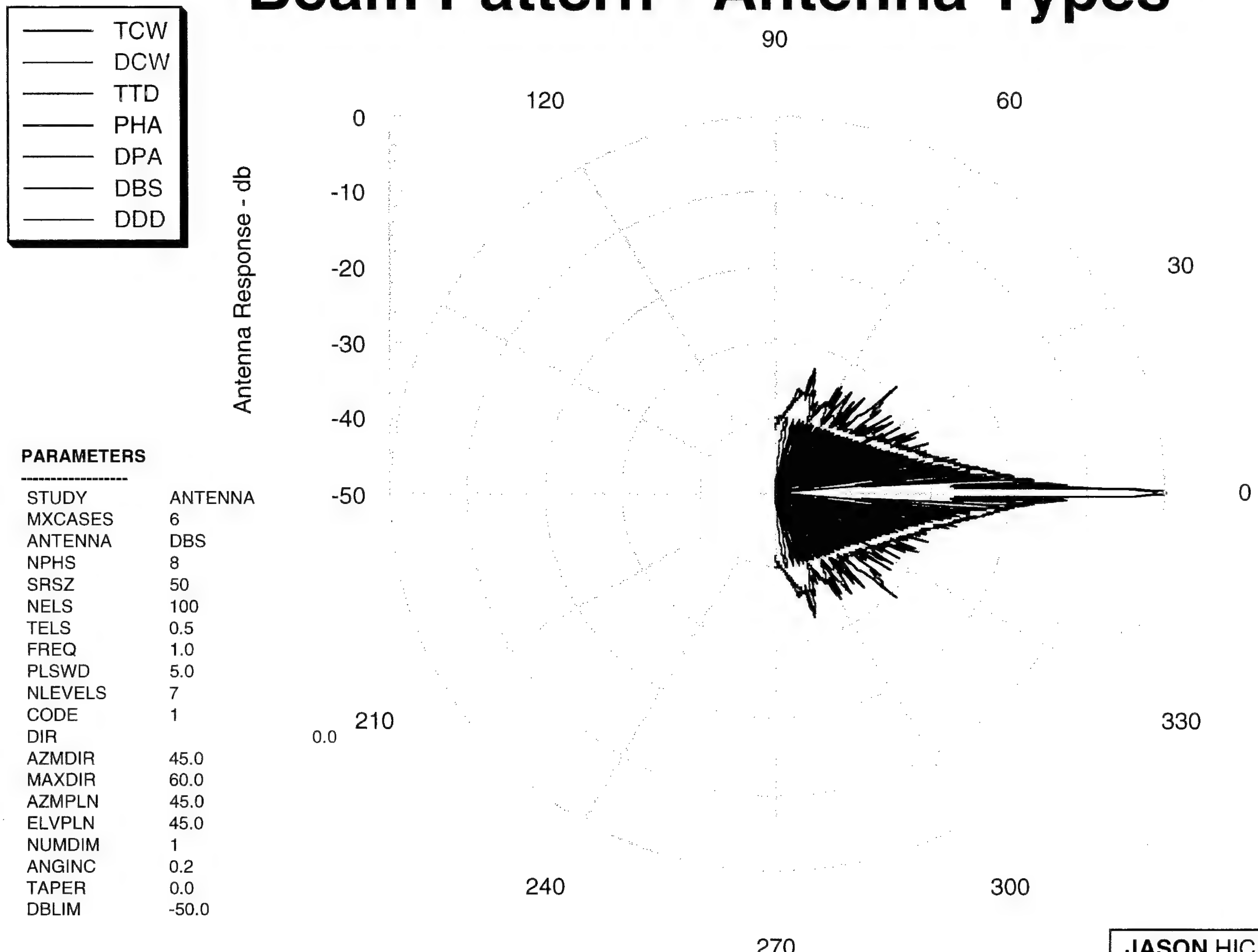
Simulation Errors

- Simulation method: Direct, time-domain trapezoidal integration of antenna element signals.
- Time-step resolution is an issue:
  - If too small, simulation cost is too high
  - If too large, simulation is not accurate.
- A number of time step sizes were tried.
- Result: 0.125 nanoseconds was chosen.
- An example at 0.25 ns is shown to validate this choice.

Simulation Errors

Computer simulation is employed to study the different beam forming schemes and their parameter settings. To get an accurate simulation the simulation time-step cannot be too big. For the range of parameters of interest, a simulated time step of 0.125 nanoseconds is the right choice. This will be shown in a later plot.

Beam Pattern - Antenna Types



PARAMETERS

STUDY	ANTENNA
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	100
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	1
ANGINC	0.2
TAPER	0.0
DBLIM	-50.0

JASON HICAPOR

Beam Pattern - Antenna Types

This plot shows a nominal beam steered to zero degrees for each of the seven beam forming methods studied. These are:

TCW: True Carrier Wave beam forming.

When an RF carrier is modulated, with a pulse for example, the resulting signal has a wider bandwidth than the original carrier. As a result beams formed on a pulsed modulated carrier have additional artifacts (side-lobes, etc..) TCW is thus shown so comparisons can be made.

DCW: Discrete Carrier Wave beam forming.

If the phases of each carrier wave at each antenna element are set in discrete steps, artifacts can appear in the beam. Simulation of the DCW provides a measure of the required number of discrete phases needed for accurate beam forming.

PHA: Phased Array, pulsed RADAR beam forming.

This is the classic phased array RADAR with continuous setting of each phase at each antenna element.

DPA: Discrete Phased Array beam forming.

This is the classic PHA but with discrete settings of each phase angle.

Beam Pattern - Antenna Types
(concluded)

<u>TTD: True Time Delay beam forming.</u>

Beams are formed by first pulse modulating a carrier, then delaying a copy of it for each antenna element. Beam steering is accomplished by providing the correct delay for each element.

<u>DBS: Digital Beam Synthesis beam forming.</u>

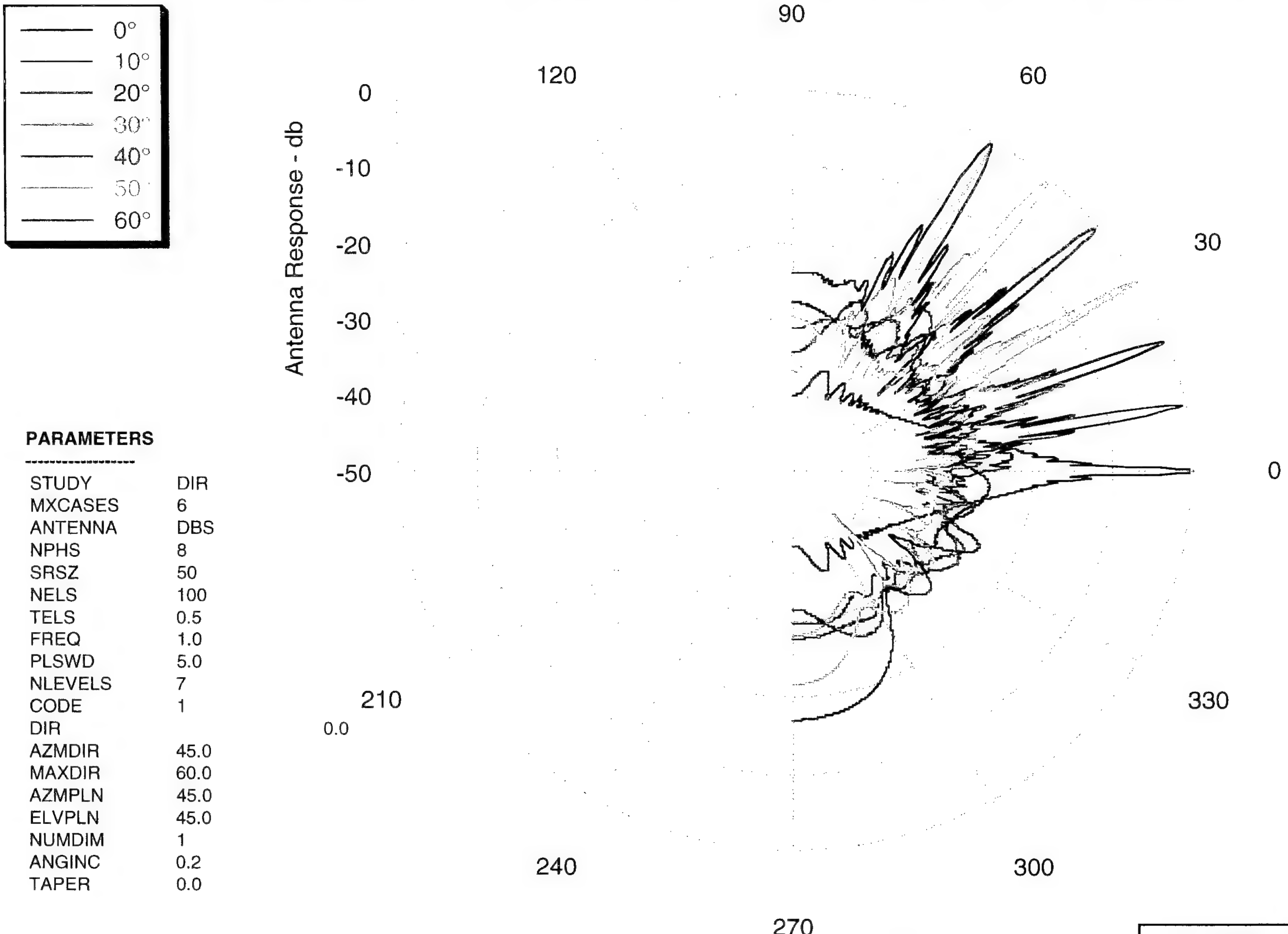
This is beam forming by separately modulating an RF carrier at each element. Time delay is controlled by a digital shift register for each element. Hence, time delays are discrete.

<u>DDD: Digital-Carrier and Digital Delay, beam forming.</u>

It may be desirable to employ digitally synthesized carrier waves in RADAR systems. DDD provides various discrete approximations to sine waves to test the DBS system on these types of carriers.

This page left blank intentionally.

Beam Pattern - Steered Elevation Beams



PARAMETERS

STUDY	DIR
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	100
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	1
ANGINC	0.2
TAPER	0.0

JASON HICAPOR

Beam Pattern - Steered Elevation Beams

This antenna response plot illustrates the computer simulation of the base-line case. All the other plots are variants that explore different issues with the DBS scheme.

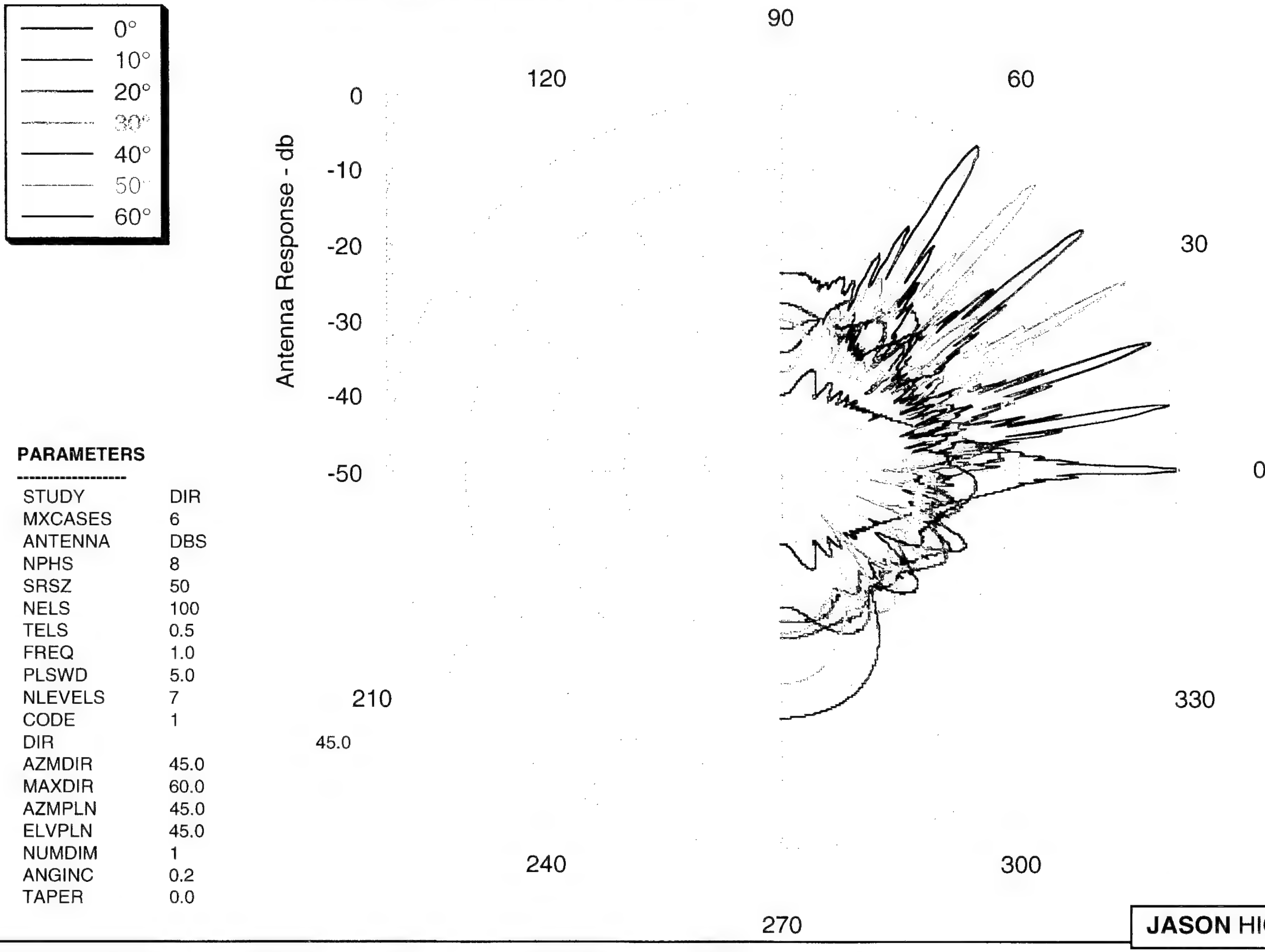
The legend for the individual plots is shown in the upper left-hand side of the figure. For this figure, the individual plots are for beams steered to point to elevation angles of 0°, 10°, 20°, 30°, 40°, 50°, and 60°.

The parameter settings for this base- line case are shown on the lower left-hand side of the figure.

For this base-line case, a simple pulse is modeled. The carrier frequency is 1 GHz, the antenna is a 1-D linear array of 100 elements spaced at $\lambda/2$ (0.5 ns), there is no tapering, and the DBS employs shift registers of length 50 and 8 phase angles.

Beam Pattern -Steered Elevation Beams

[Time Resolution : 0.25]



PARAMETERS

STUDY	DIR
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	100
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	1
ANGINC	0.2
TAPER	0.0

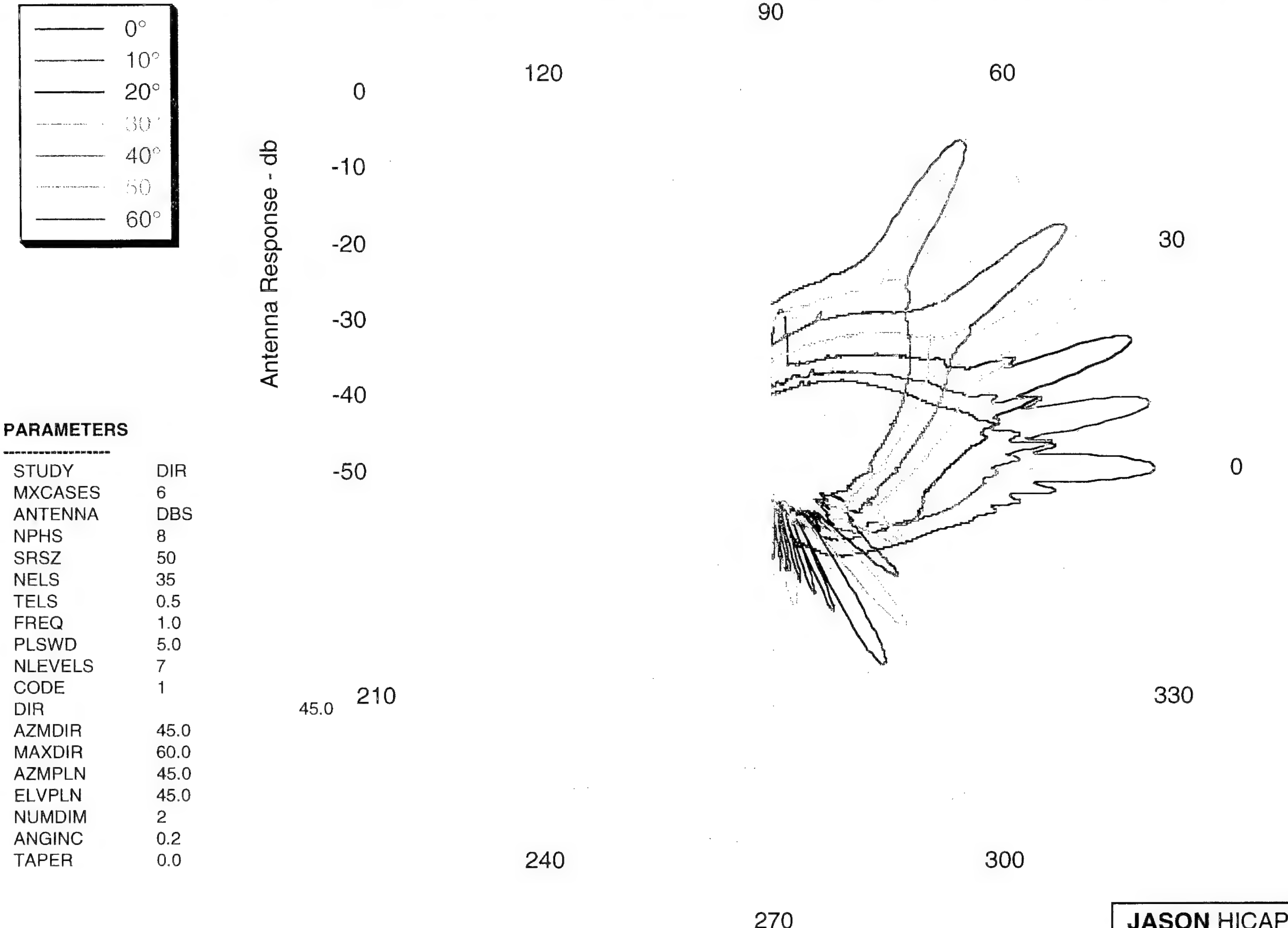
JASON HICAPOR

Beam Pattern - Steered Elevation Beams
[Time Resolution: 0.25]

The simulation method employed is direct time-domain trapezoidal integration of all the signals produced by all the antenna elements.

An issue then is the resolution of the numeric integration. This figure shows the results of employing a time-step of 0.25 nanoseconds (ns); that is, twice as long as the step of 0.125 ns employed in all the other studies. It can be seen that very little difference results. If the time-step is increased to 0.5 nsec, noticeable artifacts appear. (This result is not shown.) The conclusion is that 0.125 nsec is the proper time step to employ in the calculations.

Beam Pattern - Steered Elevation Beams [2-D]



PARAMETERS

STUDY	DIR
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	35
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	2
ANGINC	0.2
TAPER	0.0

JASON HICAPOR

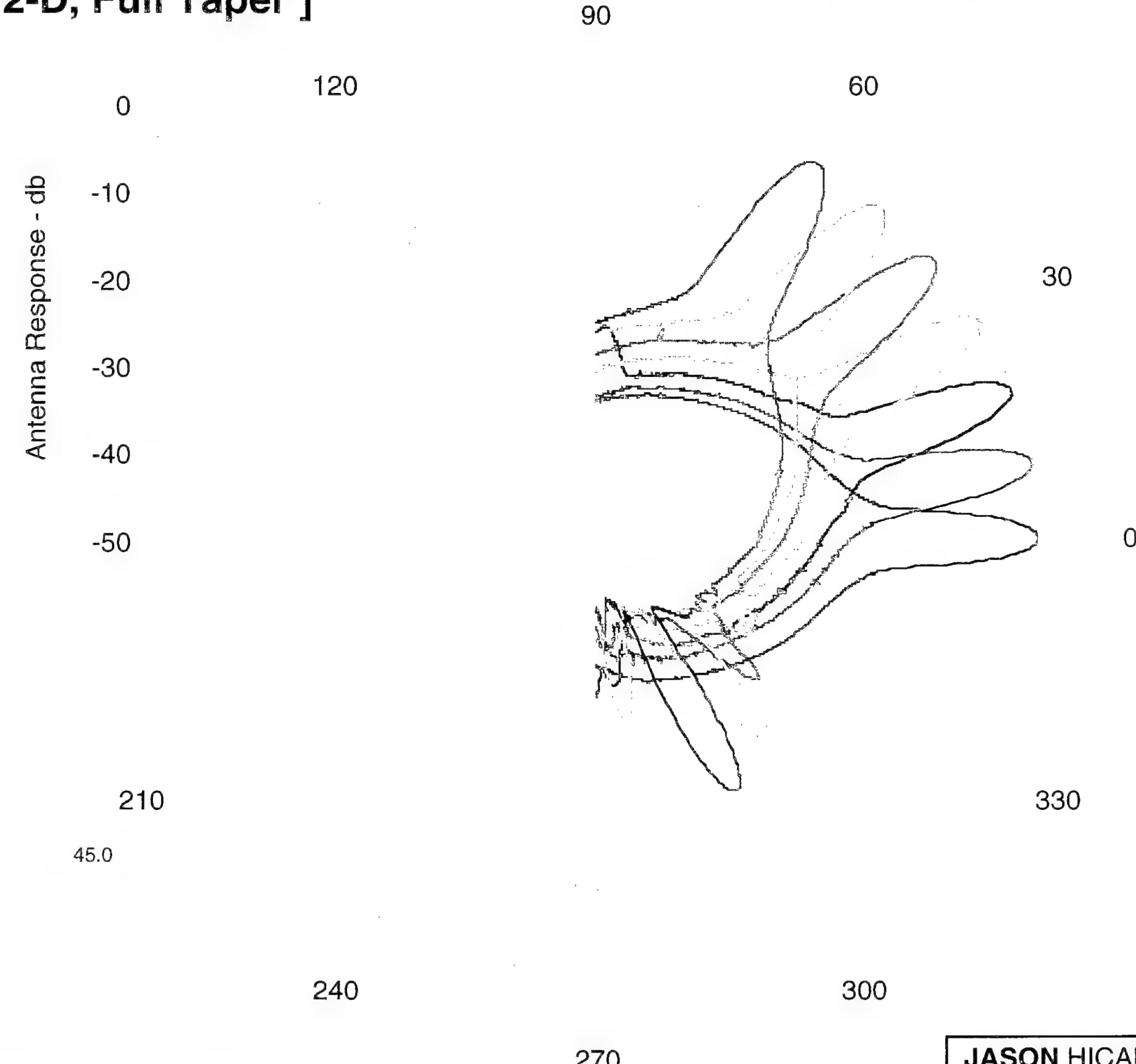
Beam Pattern - Steered Elevation Beams [2-D]

This plot is for a two dimensional antenna of 35 by 35 elements spaced at $\lambda/2$. This shows beams steered to elevation angles from 0° to 60°. The relatively large side lobes at 300° result from the steered azirmuth angle being set to 45°, and the elevation angle to 60°.

Beam Pattern - Steered Elevation Beams

[2-D, Full Taper]

- 0°
- 10°
- 20°
- 30°
- 40°
- 50
- 60°



PARAMETERS

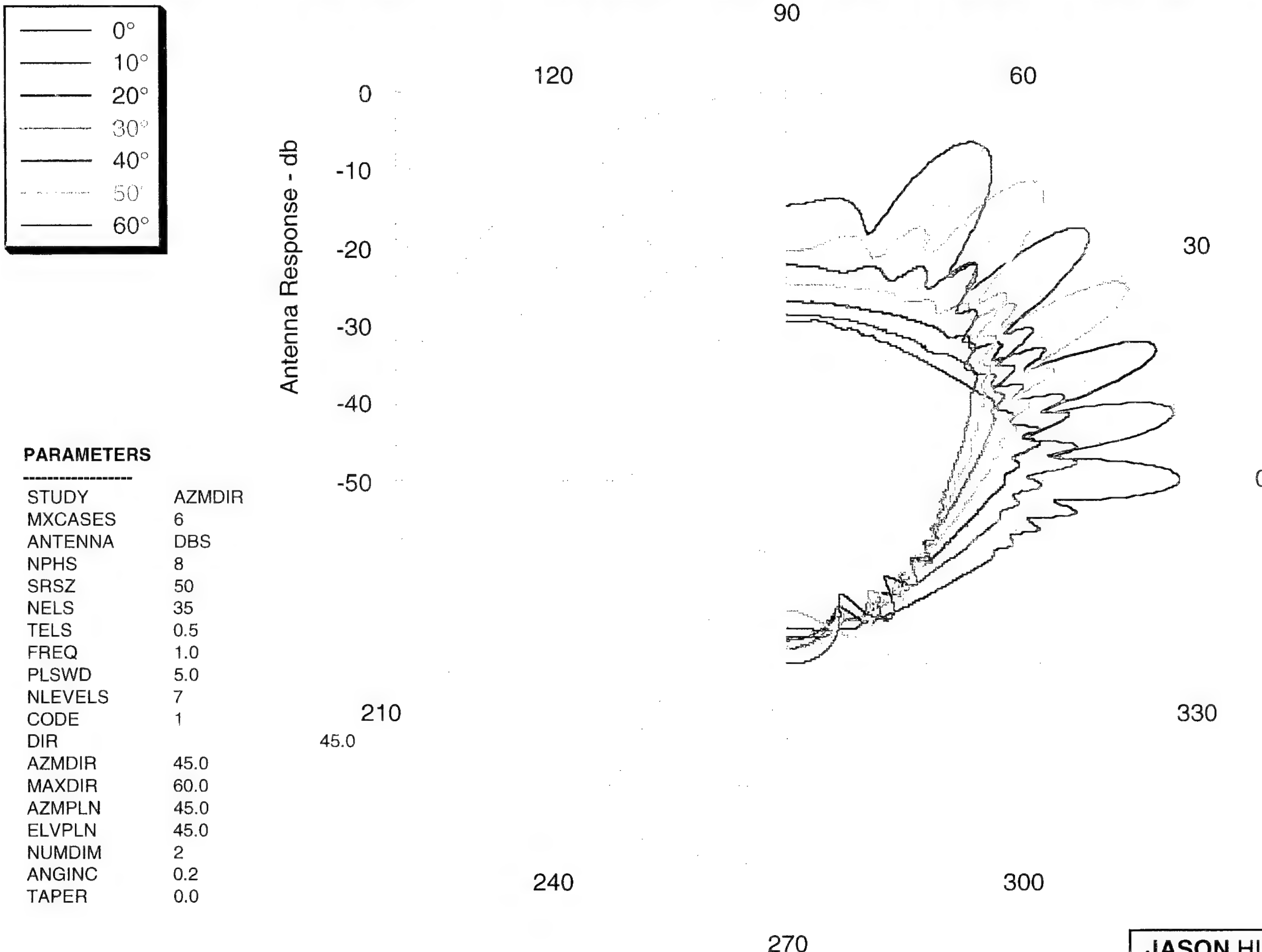
STUDY	DIR
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	35
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	45.0
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	2
ANGINC	0.2
TAPER	1.0

JASON HICAPOR

Beam Pattern - Steered Elevation Beams
[2-D, Full Taper]

This shows what tapering can do for the close-in side lobes. It does suppress them at the cost of a wider main beam. Later the amount of tapering will be varied to better illustrate this. Note that at extreme directions (e.g. 300°) and extreme steering (60°) the side lobes are not suppressed, they are enhanced! Simple amptitude tapering will not suppress these side lobes.

Beam Pattern - Steered Azimuth [2-D]



PARAMETERS

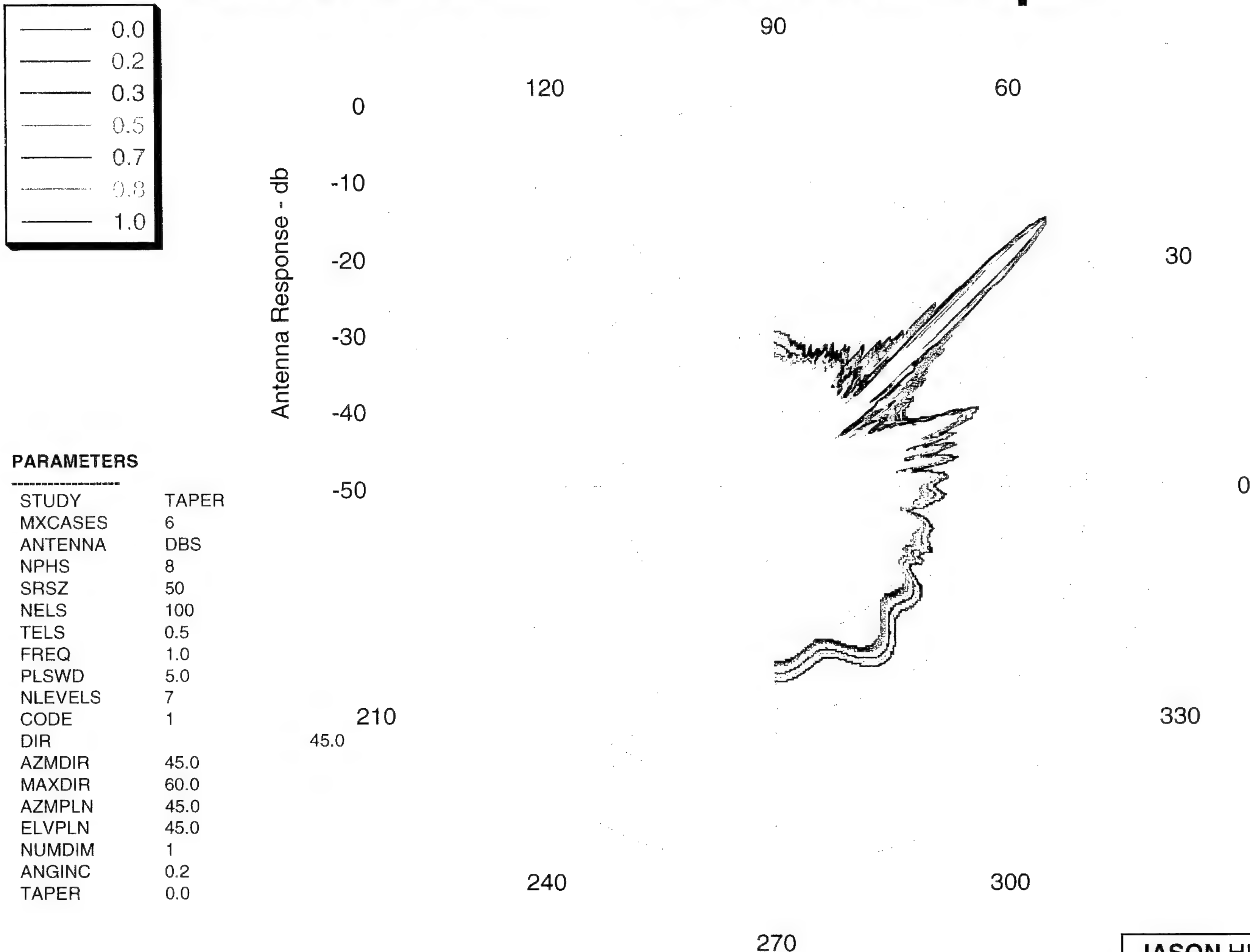
STUDY	AZMDIR
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	35
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	2
ANGINC	0.2
TAPER	0.0

JASON HICAPOR

Beam Pattern - Steered Azmiuth [2-D]

This plot is for a two dimensional antenna of 35 to 35 elements spaced at $\lambda/2$. The elevation angle is set to 45°. The beam is then steered through 0° to 60° in azimuth. Note the beam broadening due to both the 45°elevation and various azimuth steerings.

Beam Pattern - Antenna Taper



PARAMETERS

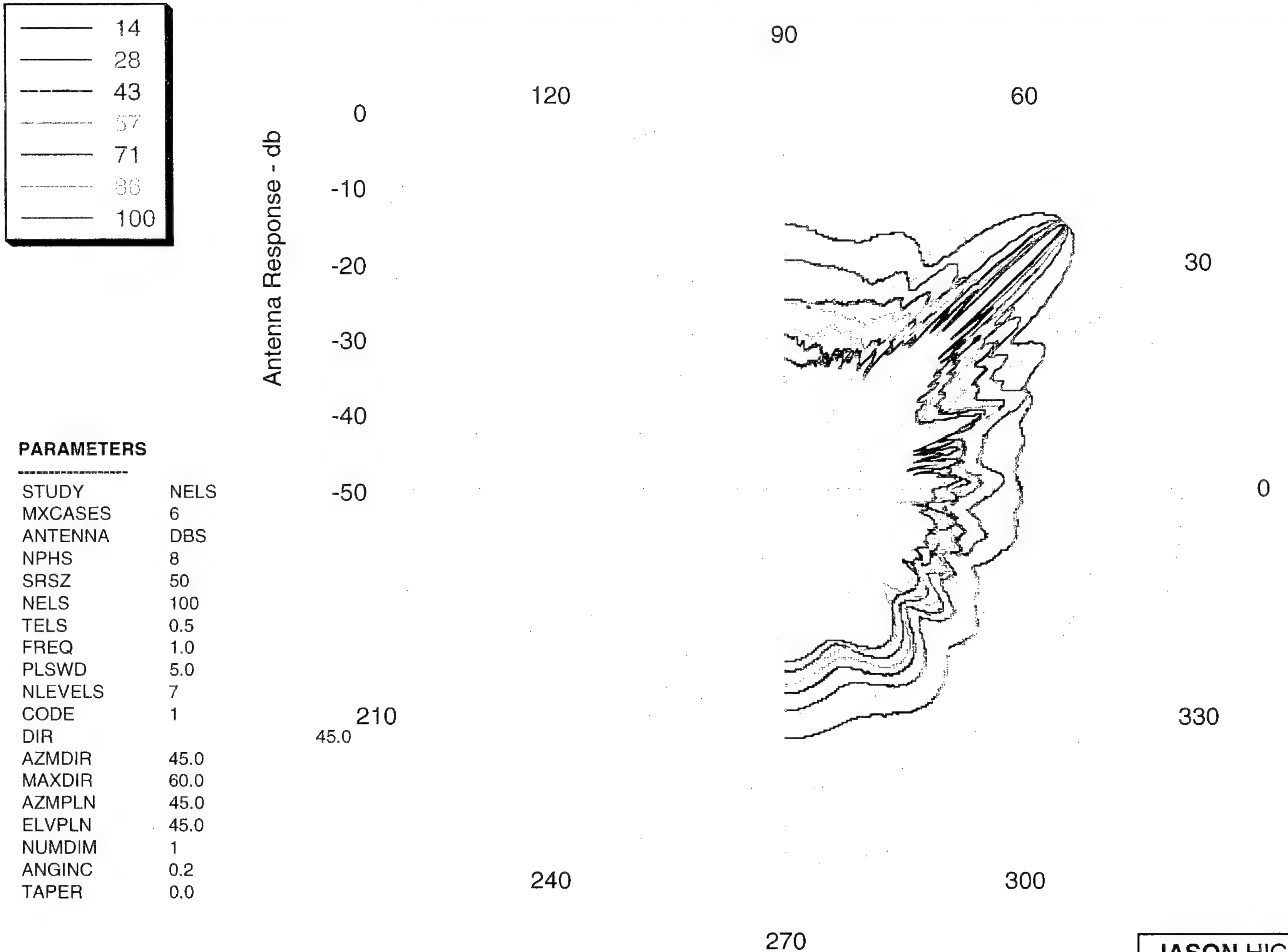
STUDY	TAPER
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	100
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	1
ANGINC	0.2
TAPER	0.0

JASON HICAPOR

Beam Pattern - Antenna Taper

Tapering the antenna array reduces the antenna side-lobes at the cost of widening the main beam. This figure shows no tapering and six uniform levels of tapering. The full tapering is a 100% raised cosine function.

Beam Pattern - Number of Elements/Dimension



PARAMETERS

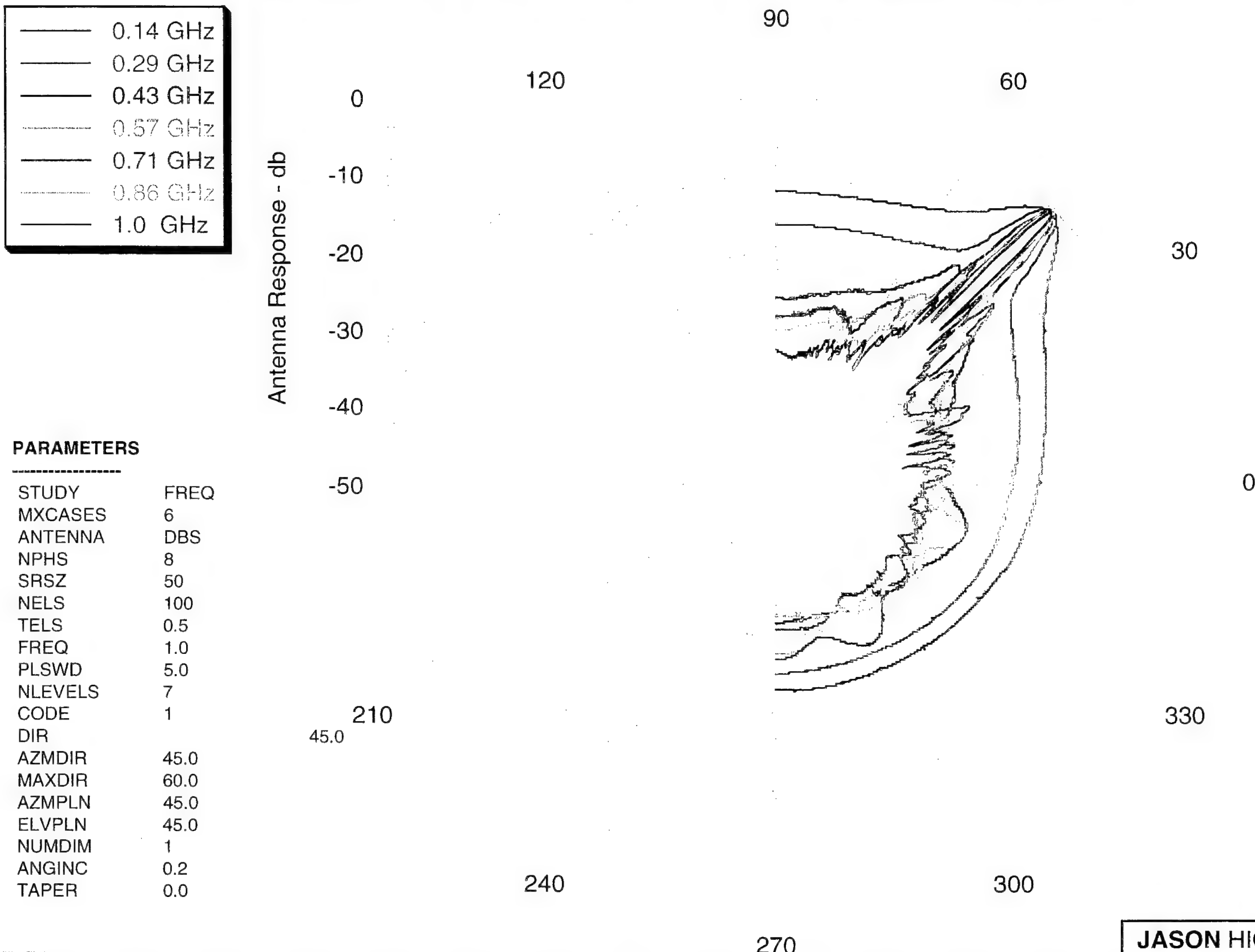
STUDY	NELS
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	100
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	1
ANGINC	0.2
TAPER	0.0

JASON HICAPOR

Beam Pattern - Number of Elements Dimension

Here the number of antenna elements is varied from 14 to 100 elements in 7 steps.
With only 14 elements, the beam is wide and the side lobes are large.

Beam Pattern - Frequencies



PARAMETERS

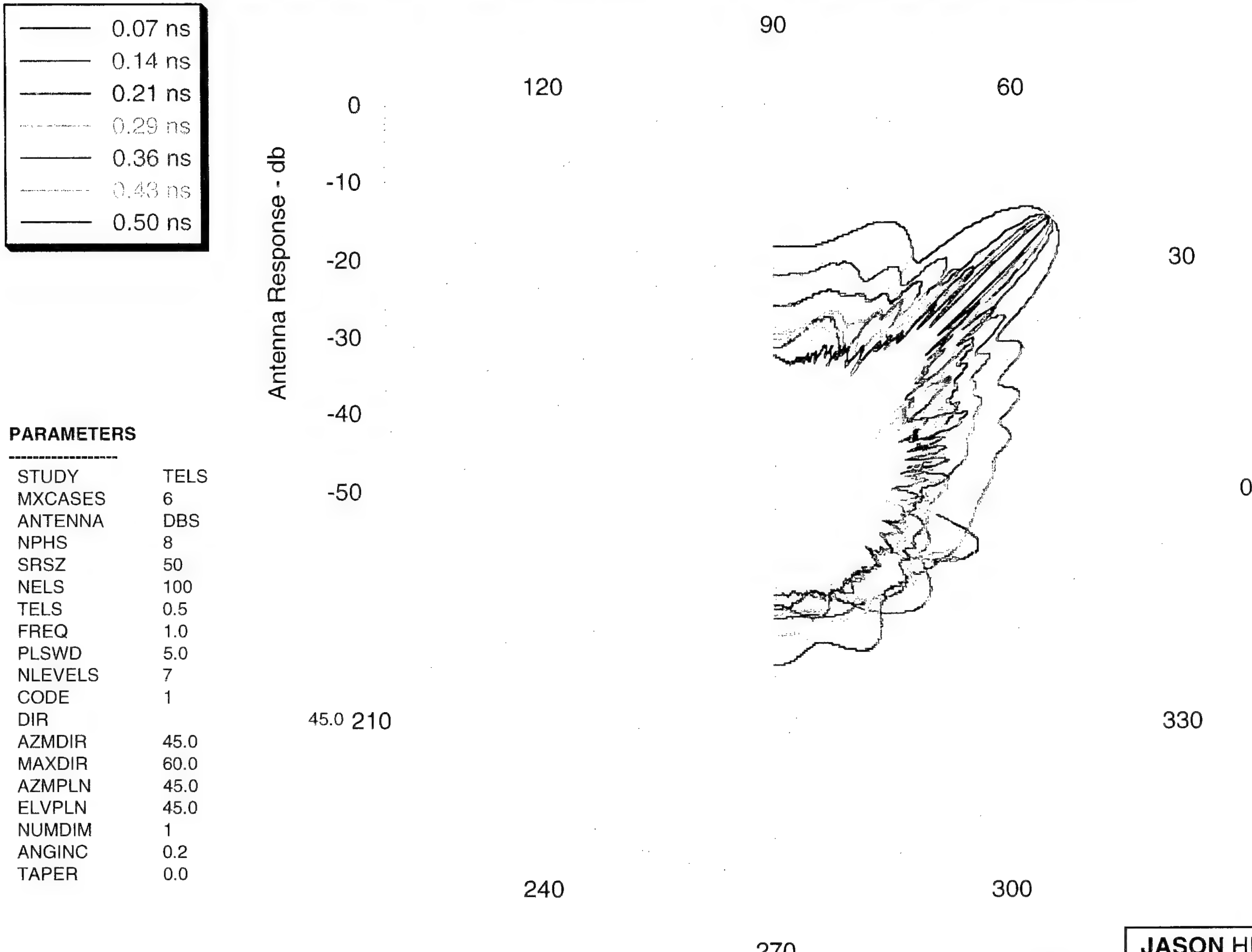
STUDY	FREQ
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	100
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	1
ANGINC	0.2
TAPER	0.0

JASON HICAPOR

Beam Pattern - Frequencies

This illustrates operating the antenna at lower frequencies. The beam broadens and side lobes rise as expected as the frequency decreases.

Beam Pattern - Time/Element



PARAMETERS

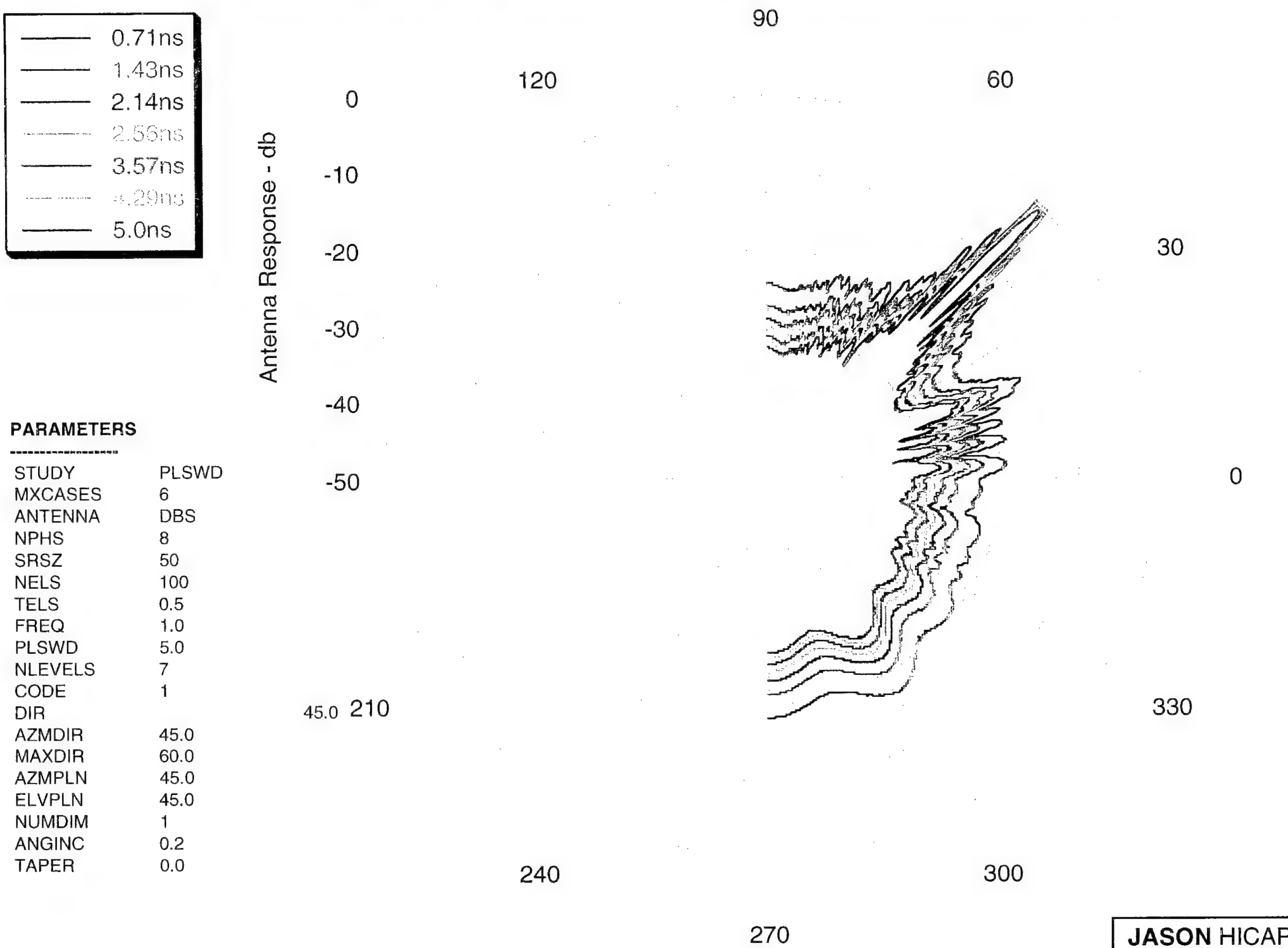
STUDY	TELS
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	100
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	1
ANGINC	0.2
TAPER	0.0

JASON HICAPOR

Beam Pattern - Time/Element

The spacing of the 100 antenna elements is varied from 0.07 nsec to 0.50 nsec in 7 steps. The 0.50 nsec spacing corresponds to $\lambda/2$ spacing. It can be seen that as the antenna gets smaller (smaller spacing), the beam widens and side-lobes increase as expected.

Beam Pattern - Pulse Widths



PARAMETERS

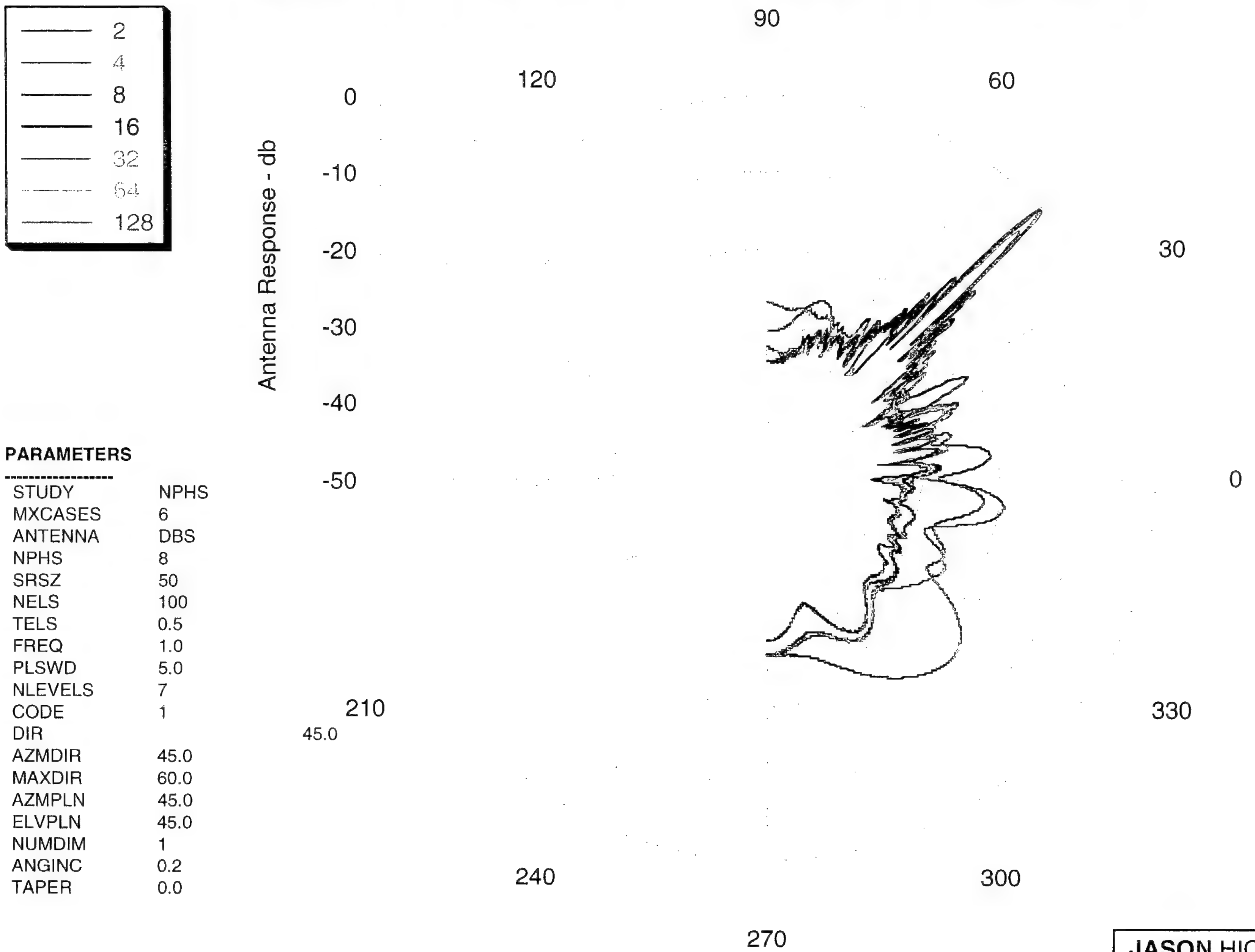
STUDY	PLSWD
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	100
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	1
ANGINC	0.2
TAPER	0.0

JASON HICAPOR

Beam Pattern - Pulse Widths

In beam forming for pulses, the shorter the pulse the wider the spectrum of the signal. Thus, the more difficult the beam forming problem. This plot shows the beams when the pulse width is shortened in seven steps from 5.0 nsec to 0.7 nsec. Short pulses cause high side lobes.

Beam Pattern - Fixed Phases



PARAMETERS

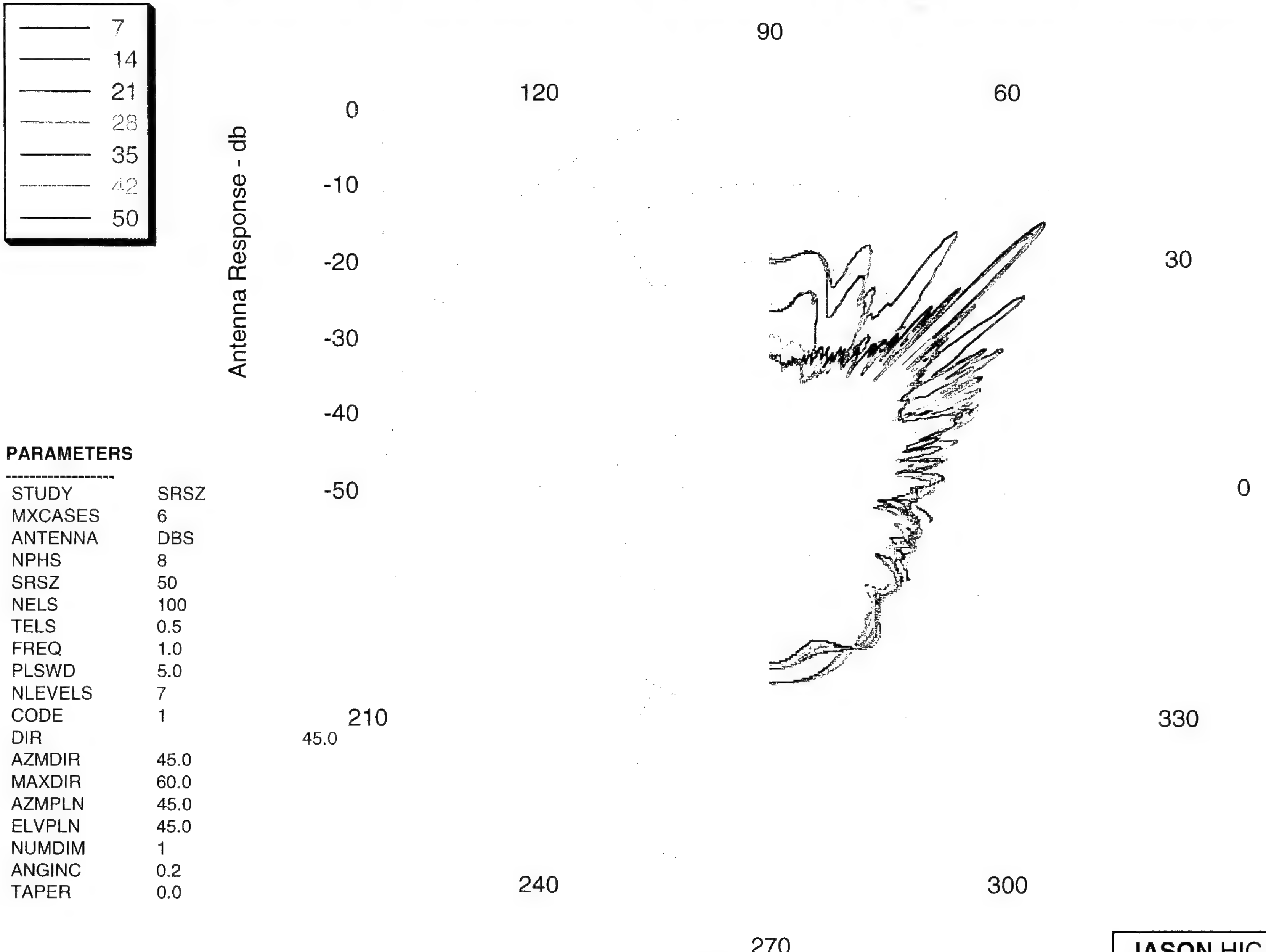
STUDY	NPHS
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	100
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	1
ANGINC	0.2
TAPER	0.0

JASON HICAPOR

Beam Pattern - Fixed Phases

Because the Digital Beamforming System (DBS) employs discrete steps to set the pulse of each antenna element, the more steps, the better the approximation to the desired phase. If only two steps (of 0° and 180°) are used, the main beam is decreased and the side lobe response is bad. However, with at least eight steps the approximation results in excellent beamforming. More steps offer little improvement.

Beam Pattern - Shift Register Sizes



PARAMETERS

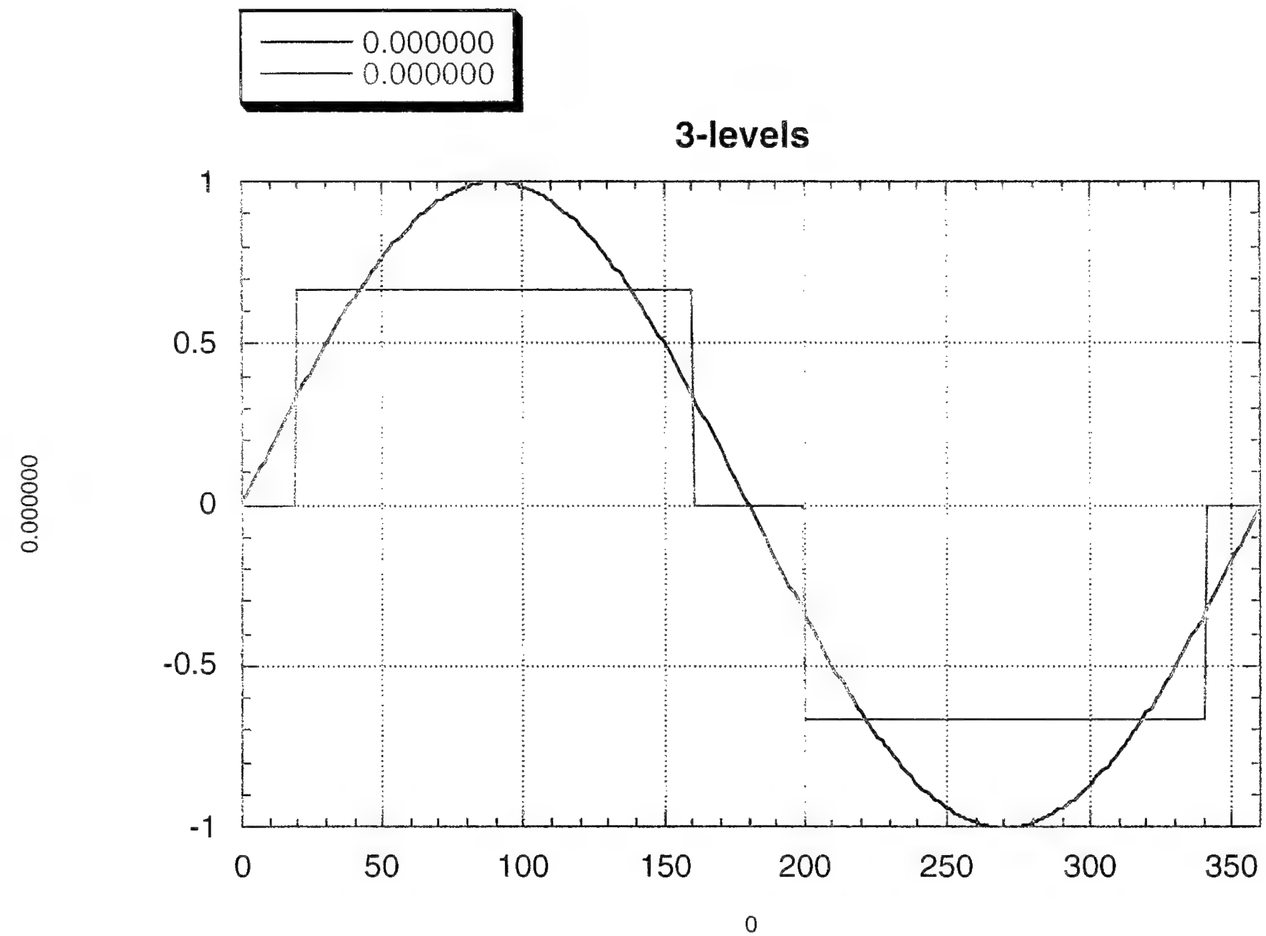
STUDY	SRSZ
MXCASES	6
ANTENNA	DBS
NPHS	8
SRSZ	50
NELS	100
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	1
ANGINC	0.2
TAPER	0.0

JASON HICAPOR

Beam Pattern - Shift Register Sizes

For the base case, a shift register length of 50 matches the 100 element antenna with $\lambda/2$ spacing. Fewer (therefore slower) register stages only give an approximation to the desired delay pattern in time. The artifacts of the approximation when the number of shift registration stages is only seven is apparent in the high-level side-lobes it generates.

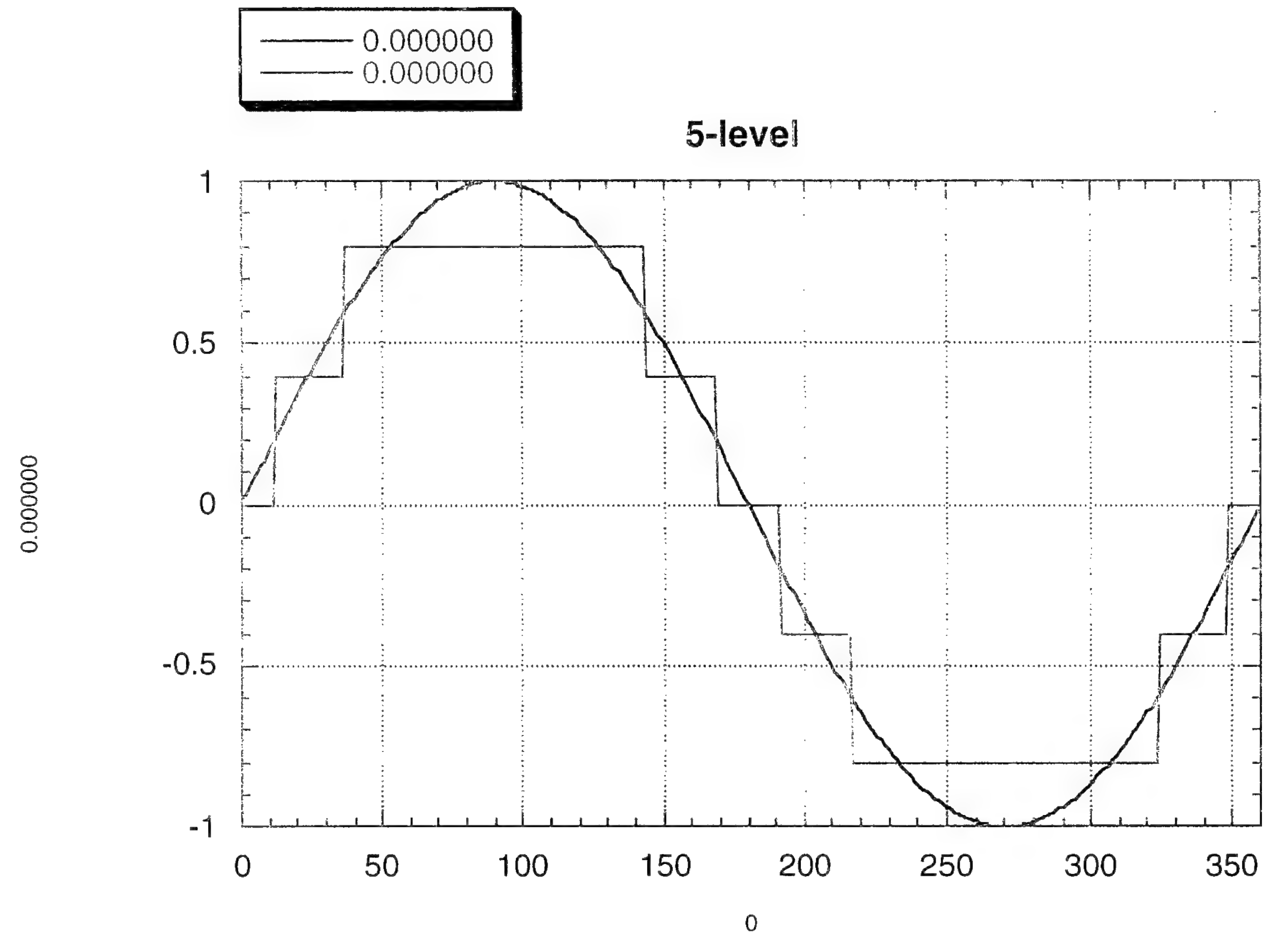
Three-Level Digital Carrier



Three-Level Digital Carrier

In all the previous cases studied, the carrier waveform was an analog sine wave. Max Yoder of ONR suggested that this carrier could also be digitally synthesized. Here is a sine wave and its three level approximation.

Five-Level Digital Carrier

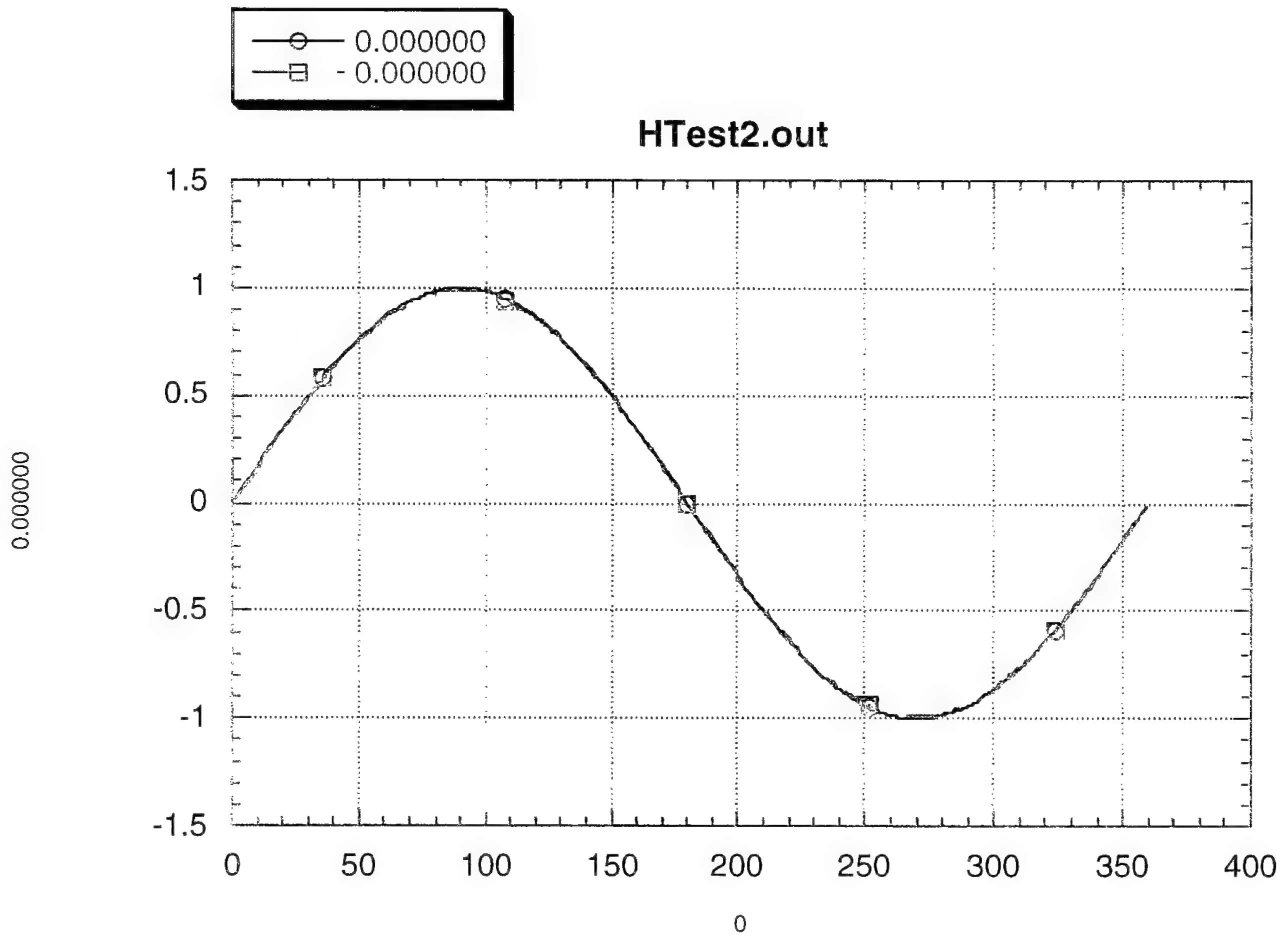


JASON HICAPOR

Five-Level Digital Carrier

This is a five-level approximation to a sine wave.

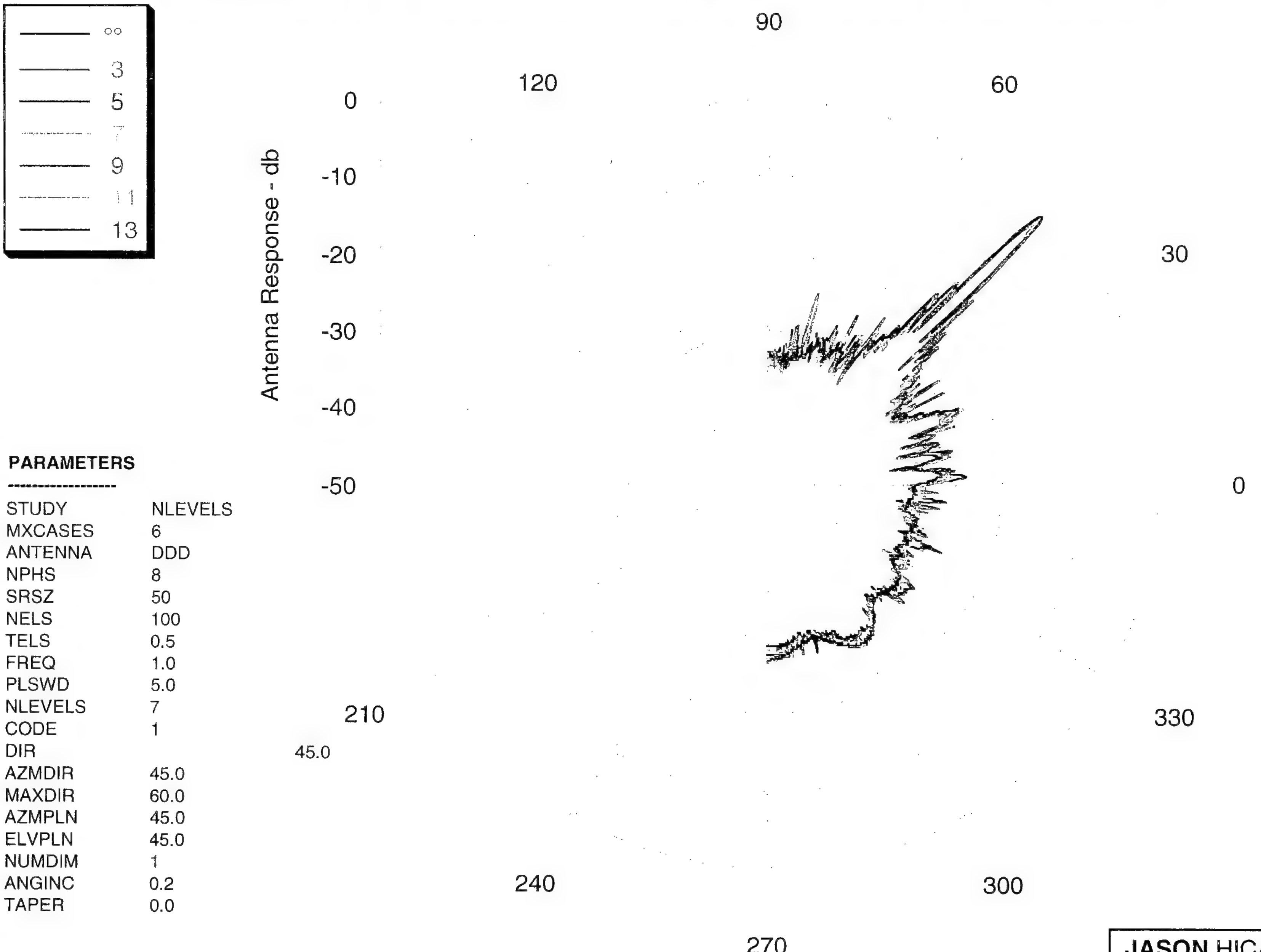
Sixty-Three-Level Digital Carrier



Sixty-Three-Level Digital Carrier

This is a sixty-three-level approximation to a sine wave.

Beam Pattern - Discrete Levels



PARAMETERS

STUDY	NLEVELS
MXCASES	6
ANTENNA	DDD
NPHS	8
SRSZ	50
NELS	100
TELS	0.5
FREQ	1.0
PLSWD	5.0
NLEVELS	7
CODE	1
DIR	
AZMDIR	45.0
MAXDIR	60.0
AZMPLN	45.0
ELVPLN	45.0
NUMDIM	1
ANGINC	0.2
TAPER	0.0

JASON HICAPOR

Beam Pattern - Discrete Levels

Here are the beam patterns that result when the carrier is changed from an analog sine wave down to 3, 5, 7, 9, 11 and 13 step approximations to a sine wave (see previous plots.) It is interesting that for beam forming purposes, a crude (i.e. five level) approximation seems to be good enough.

Conclusions and Recommendations

- If the shift register and number of phase parameters are properly chosen, the Digital Beam Synthesis (DBS) scheme performs as well as True Time Delay (TTD) scheme.
- DBS can provide <u>VERY</u> large saving in cost and size over TTD with the <u>SAME</u> performance.
- A vigorous experimental test of DBS should be immediately started.
- DBS could radically impact ship, aircraft and ground-base RADAR systems in terms of lower cost and smaller size.

Conclusions and Recommendations

The simulations carried out for this presentation show that modest approximation does not hurt beam forming. This means that modest cost Digital Beam System (DBS) can lead to large savings in weight and cost in phased array antenna systems, with essentially no loss in performance.

Because of the very large potential weight, size and cost savings, it is recommended that ONR proceed to a vigorous experimental construction and test of a DBS antenna.

This page left blank intentionally.

APPENDIX
SIMULATION PROGRAM

Appendix Simulation Program

This simulation program is written in standard C programming language and employs only the standard ANSI libraries. It should compile and run on any computer equipped with a C compiler. There is no input. Various runs result from changing the parameters in the define statements at the top of the program

The output is a tabular file suitable for plotting by any standard plot program. The Kaleidagraph plotting package for the MAC produced the plots shown here.

```
/* main.c */

/* ----------------------------------------------------------------------------------------- */
#define STUDY    "DIR"   /* Define desired study,i.e. What varies.. in study.Choices: */
                           /* BASE,DIR,ANTENNA,NPHS,SRSZ,NELS,TELS,FREQ,PLSWD,NLEVELS,     */
                           /* CODE,AZMDIR,TAPER                                              */
#define MXCASES  6         /* No. of cases  for  studies: (0,MXCASES)                        */
#define ANTENNA  "DBS"     /* Antenna Type for BASE study:(TCW,DCW,PHA,DPA,TTD,DBS,DDD)      */
#define NPHS     8         /* Number of discrete phases                                      */
#define SRSZ     50        /* Number of shift register stages per antenna element            */
#define NELS     100       /* Number of ant elements (in each direction if 2-D array)        */
#define TELS     0.5       /* Time (ns) between antenna elements                             */
#define FREQ     1.0       /* Frequency of carrier in GHz                                    */
#define PLSWD    5.0       /* Transmitter Pulsewidth in ns                                   */
#define NLEVELS  7         /* No. of levels for discrete sin funct. (= 1 for cont. sin)      */
#define CODE     1         /* Pulse Compression Code. 114 = Barker 7 bit code                */
#define DIR      45.0      /* Steered beam elevation in degrees                              */
#define AZMDIR   45.0      /* Steered azimuth direction in degrees for 2-D array             */
#define MAXDIR   60.0      /* Maximum Angle to be steered in degrees in DIR study            */
#define AZMPLN   45.0      /* Azimuth Angle for observation plane (in degrees)               */
#define ELVPLN   45.0      /* Elevation Angle for observation plane (in degrees)             */
#define NUMDIM   1         /* Number of dimensions for the antenna array                     */
#define ANGINC   0.2       /* Angle Increment of plot data in degrees                        */
#define TAPER    0.0       /* Antenna taper: 0.0 = none; 1.0 = full raised cosine taper.     */
#define DBLIM    -50.0     /* dB Limit to minimum of graph plot                              */
/* ----------------------------------------------------------------------------------------- */

#define TINC     0.125     /* Time slice calculation resolution in ns */
#define ENORM    0.5       /* Energy Normalization factor-- Half-wave rectifier  */
#define TWOPI    6.2831853 /*  2π */

#include<stdio.h>
#include <stdlib.h>
#include<math.h>

FILE *fp;

float ff;
int ant,dd,study;
long code,i,ir,isr,j,jr,k,l,m,n,nels,nl,nlevels,nphs,nsrs;
double a, ang, azmdir,azmpln,b,c,d,db,dir,dt,e,elvpln,em,freq,idir,p,pd,pl,plswd;
double pn,phir,rd,s,sf,sp,t,ta,taa,tb,taper,td,tels,tdir,tt,tp,tmin,tmax,tx,ty,w,x,y,z;
double dirinc, in,nn,nt,nf,np;

char *file_name[] =   {
    "nul",
    "Steered Elev.'s",
    "Antenna Types",
    "Fixed Phases",
    "Shft Reg. Sizes",
    "No.Elements/Dim",
    "Time / Element",
    "Frequencies",
    "Pulse Widths",
    "Discrete Levels",
    "Types of Code",
    "Steered Azimuth",
    "Antenna Taper"    };
```

```
/* ----------------------------------------------------------------------- */



/* ----------------------------------------------------------------------- */


void base_study(int);
void dir_study(int);
void studies(int, int);
void cw(int);
long code_tbl( long );
long code_length(long );
unsigned hash(char *,unsigned,unsigned);
double rcos(long );
double dsin(double);
double discrete_phase(long );
double discrete_delay(long );
double d_phase(double );
double d_delay(double );
double pulse( double );
double coded_pulse( double,long );
double time_a(long );
double time_b(long );
double time_bb(long );
double cw_power(int);
double pulse_energy(int);
double sum_eles(int,double,double,double,double,double);
void init_parameters(void);
void direction_parameters(double,double);
void ant_file_open(char *);
void study_file_open(char *);
void inc_calc(int);
void dbprint(double);
void inc_print(double);
void base_header(void);
void header_print(double);
void note_print(void);
void head_print(int);
void line_print(int);
void para_print(int);
void note_write(void);
void study_parm_write(char *);
void para_write(int);


/* ----------------------------------------------------------------------- */



/* ----------------------------------------------------------------------- */
```

```
void main(void)
{
  nphs   = NPHS;
  nsrs   = SRSZ;
  nels   = NELS;
  tels   = TELS;
  freq   = FREQ;
  plswd = PLSWD;
  idir    = DIR;
  nlevels = NLEVELS;
  code    = CODE;
  azmdir = AZMDIR;
  taper = TAPER;
  elvpln = ELVPLN;        /* fixes a compiler bug */
  azmpln = AZMPLN;

  dirinc = MAXDIR/MXCASES;
  in = ((double)SRSZ)/(1+MXCASES);
  nn = ((double)NELS)/(1+MXCASES);
  nt = TELS/(1+MXCASES);
  nf = FREQ/(1+MXCASES);
  np = PLSWD/(1+MXCASES);

  rd = TWOPI/360.0;
  taa = tels*sin(rd*idir);
  jr = (90.0/ANGINC);
  nl = 1 + (NUMDIM - 1)*(nels -1);
  init_parameters();

  i = hash(ANTENNA,2,13);
  switch(i)                    /* Hash table for type of antenna */
    {
        case 0  : ant = 3; break;       /* TTD */
        case 3  : ant = 5; break;       /* DPA */
        case 6  : ant = 6; break;       /* DBS */
        case 8  : ant = 7; break;       /* DDD */
        case 9  : ant = 4; break;       /* PHA */
        case 11 : ant = 1; break;       /* TCW */
        case 12 : ant = 2; break;       /* DCW */
        default : printf(" Unknown type antenna specified in main. Antenna = %s\n",ANTENNA);
    }
 /* printf("Type antenna = %s; Hashval = %d; Value of ant = %d \n",ANTENNA,i,ant);return; *
 printf("-----------------------------------------------\n");
 i = hash(STUDY,22,27);study = 0;
 switch(i)
   {
        case 0  :  studies(ant,6);      break;      /* TELS study       */
        case 2  :  studies(ant,7);      break;      /* FREQ study       */
        case 4  :  studies(ant,2);      break;      /* ANTENNA study    */
        case 5  :  studies(ant,11);     break;      /* AZMDIR study     */
        case 7  :  studies(ant,8);      break;      /* PLSWD study      */
        case 13 :  studies(ant,1);      break;      /* DIR study        */
        case 15 :  studies(ant,9);      break;      /* NLEVELS study    */
        case 17 :  studies(ant,4);      break;      /* SRSZ study       */
        case 19 :  studies(ant,3);      break;      /* NPHS study       */
        case 21 :  studies(ant,5);      break;      /* NELS study       */
        case 22 :  base_study(ant);     break;      /* BASE study       */
        case 23 :  studies(ant,12);     break;      /* TAPER study      */
        case 25 :  studies(ant,10);     break;      /* CODE study       */
```

```
        default : printf(" Unknown study in main. Study = %s  case = %d\n", STUDY,i);
    }

/* printf("Type study = %s; Hashval = %d; Value of study = %d\n",STUDY,i,study);return; */

    return;

}               /* End of main program   */


/* ------------------------------------------------------------------------------ */




/* ------------------------------------------------------------------------------ */

void base_study(int ant)
  {
   long j;
   direction_parameters(DIR,AZMDIR);
   ant_file_open(ANTENNA);
   base_header();
   for (j = -jr; j <= jr; j++)
     {
      switch(ant)
          {
            case 1 : dbprint( cw_power( ant) ) ;     break;
            case 2 : dbprint( cw_power( ant) ) ;     break;
            case 3 : dbprint( pulse_energy(ant) );   break;
            case 4 : dbprint( pulse_energy(ant) );   break;
            case 5 : dbprint( pulse_energy(ant) );   break;
            case 6 : dbprint( pulse_energy(ant) );   break;
            case 7 : dbprint( pulse_energy(ant) );   break;
            default:  printf(" Unknown type of ant specified in dir study.\n");
          }
     }
   printf("  End of %s Data  \n",ANTENNA);
   fclose(fp);
 }          /* end of base study  */
/* ------------------------------------------------------------------------------ */

void studies(int ant, int study)
    {
     long i, n;
     study_parm_write(file_name[study]);
     study_file_open(file_name[study]);
     para_write(study);
     note_print();
     head_print(study);
     for (j = -jr; j <= jr; j++)       /* Steps through each observation angle */
      {
        ang = (long)(1000*j*ANGINC)/1000.0;
        printf("%4.1f",ang);   fprintf(fp,"%4.1f",ang);
        for (i = 0; i <= MXCASES; i++)
         {
           switch(study)
             {
              case 1 :  idir = i *dirinc;               break;  /* DIR     */
```

```
                case 2 :  ant    = i+1;                        break;  /* ANT      */
                case 3 :  nphs = 2 << i;                       break;  /* NPHS     */
                case 4 :  nsrs = (long)((i+1)*in);             break;  /* SRSZ     */
                case 5 :  nels = (long)((i+1)*nn);             break;  /* NELS     */
                case 6 :  tels = (i+1)*nt;                     break;  /* TELS     */
                case 7 :  freq   = (i+1)*nf;                   break;  /* FREQ    */
                case 8 :  plswd = (i+1)*np;                    break;  /* PLSWD    */
                case 9 :  nlevels = 2*i+1;                     break;  /* NLEVELS */
                case 10 : code   = code_tbl(i);                break;  /* CODE     */
                case 11 : azmdir = i*dirinc;                   break;  /* AZMDIR   */
                case 12 : taper = ((double)i)/MXCASES;;        break;  /* TAPER    */
                default:  printf(" Unknown type of study specified in studies.\n");
               }
            ta = tels*sin(ang*rd); /* Times for each observed direction j */
            tb = tels*sin(azmpln*rd)*cos(ang*rd);
            if(study == 11) {ta = tels*sin(elvpln*rd);
                             tb = tels*cos(elvpln*rd)*sin(ang*rd);}
                                  /* observations in elevation plane */
            init_parameters();
            direction_parameters(idir,azmdir);
            inc_calc(ant);
         }  /* end of study case loop */
       printf("\n");  fprintf(fp,"\n");
     }   /* end of angle (j) loop  */
   printf("---------------------------------------------------------------------\n");
   printf("  End of %s Data  \n",ANTENNA);
   fclose(fp);

} /* end of studies  */


/* ------------------------------------------------------------------------------- */

void inc_calc(int ant)
  {
          switch(ant)
             {
                 case 1 : inc_print( cw_power( ant) ) ;   break;
                 case 2 : inc_print( cw_power( ant) ) ;   break;
                 case 3 : inc_print( pulse_energy(ant) );   break;
                 case 4 : inc_print( pulse_energy(ant) );   break;
                 case 5 : inc_print( pulse_energy(ant) );   break;
                 case 6 : inc_print( pulse_energy(ant) );   break;
                 case 7 : inc_print( pulse_energy(ant) );   break;
                 default:  printf(" Unknown ant specified in inc_calc. ant=\n",ant);
             }
           return;
  }


double cw_power(int ant)
  { long k,l;
    double ltb,lty,c,s,p,rc,rcl,t;
    c = 0.0; s = 0.0;
    for (l=0; l < nl; l++)
     { ltb = l*tb;
       lty = l*ty;
       rcl = 1; if(NUMDIM >1) rcl = rcos(l);
       for (k=0; k < nels; k++)
        { t = -ltb - k*ta;  /* Observation Direction Times */
```

```
          p =  lty + k*tx;  /* Steering    Direction Times */
          switch(ant)
            {    case 1  : p = w*(t + p) ;            break;      /* TCW */
                 case 2  : p = w*t + d_phase(p) ;     break;      /* DCW */
                 default : printf(" Unknown ant = %d in cw_power\n", ant);return;
            }
          if( taper > 0.0 )
               {  rc = rcl*rcos(k);  c = c + rc*cos(p);  s = s + rc*sin(p); }
          else {   c = c + cos(p);   s = s + sin(p); }
         }
      }
      return( (c*c + s*s) / (nl*nl*nels*nels) );
   }


double pulse_energy(int ant)
 { double r, s, t ;
   s = 0.0;
   for (t=tmin;t<tmax;t=t+TINC)
    {  r = sum_eles(ant,t,ta,tb,tx,ty);
       s = s + r*r;
    }               /* end of t loop */
   return(s/em);
 }


double sum_eles(int ant,double tc,double ta,double tb, double tx, double ty)
 {      /* calculates sum of  voltage from array of antenna elements */
    double ltb, lty, p, r, rcl,t,time, z;
    long k,l;
    r=0.0;
    for(l=0;l<nl;l++)
     { ltb = tc - l*tb;
       lty =      l*ty;
       rcl = 1; if(NUMDIM >1) rcl = rcos(l);
       for (k=0; k < nels; k++)
        { t = ltb - k*ta;   /* Observation Direction Times */
          p = lty + k*tx;   /* Steering    Direction Times */
          switch(ant)
          {
           case 1 : printf(" Unknown ant = %d in pulse eval section\n",ant);return;
           case 2 : printf(" Unknown ant = %d in pulse eval section\n",ant);return;
/* TTD */  case 3 : z = sin(w*(t + p));              time = t + p;             break;
/* PHA */  case 4 : z = sin(w*(t + p));              time = t;                 break;
/* DPA */  case 5 : z = sin(w*t + d_phase(p));       time = t;                 break;
/* DBS */  case 6 : z = sin(w*t + d_phase(p));       time = t + d_delay(p);    break;
/* DDD */  case 7 : z =dsin(w*t + d_phase(p));       time = t + d_delay(p);    break;
           default : printf("Unknown ant = %d in pulse eval section\n",ant);return;
          }
         if( taper > 0.0 ) r = r + rcl*rcos(k)*coded_pulse(time,code)*z;
         else    r = r + coded_pulse(time,code)*z;
       }                                  /* end of k loop  */
    }                                    /* end of l loop  */
    return(r);
 }                                 /* end of sum_eles function  */


/* ------------------------------------------------------------------------------------------ */


/* ------------------------------------------------------------------------------------------ */
```

```
double rcos(long k){return(1.0+taper*cos((double)(k-nels/2)*TWOPI/nels));} /*taper function*

double dsin(double phi) { if(nlevels <= 1 ) return(sin(phi));  /* Discrete sin function */
  return((((double)((long)((sin(phi)+1.0)*0.499999*nlevels))*2.0)/nlevels)-1.0+1.0/nlevels);

double d_phase(double p) {return((TWOPI*((long)(p*nphs*freq+0.5) % nphs))/nphs);}

double d_delay(double p){ return(sp*(long)(0.5+p/sp));} /* sp = nels*tels/nsrs */

long code_length(long code){ long i; for(i=0;code>0;i++)code = code >> 1; return(i);}



double discrete_phase(long k) /* {return(((TWOPI*(long)((long)(k*pl+0.5) % nphs))/nphs));} *
    {
       double p,x;
       long i,m;
       x = k*pl + 0.5  ;
       m = x;
       /* nphs = NPHS; */
       i = m % nphs ;    dd = i;                 /* Selection of element phase index */
       p = ((TWOPI*i)/nphs) ;                    /* Set phase for each element..     */
       /*     p =   k * pd;           */
       return(p);
     }


double discrete_delay(long k)        /* { return(sp*(long)(0.5+pn*k));} */
    {
      double x;
      long i;
                       /*    k*TELS*sin(dir) = time delay needed to steer array   */
                       /*    y = nsrs;    z = NELS;  pn = ( y/z) * sin(dir);      */
                       /*    x = NELS*TELS; sp = x/nsrs; */ /* Shift clock period */
       x = k*pn + 0.5;
       i = x;          /* Index into shift register  */
       if(i > nsrs){printf("Error in shift register indexing.  i = %d\n",i);return;}
                       /*i = m % nsrs; */
                       /*      printf("(%d|%d), ",i,dd);              */
       x = i*sp;
       return (x);
     }

double pulse( double tt)
{ double b;
  b=0; if (tt>=0) {if(tt<plswd) b=1.0;}     /* Define RADAR Pulse */
  return(b);
}

double coded_pulse( double tt,long code)
{ long i,k;         /* Define Coded RADAR Pulse */
  i = floor(tt/PLSWD);
  if(i < 0) return(0.0);
  k = code >> i ;             /* Barker-7 Code = 114 ( 1,1,1,0,0,1,0 ) */
  if(k <= 0) return(0.0);     /* Code read right to left */
  return(2*(k % 2) -1);
}
```

```
void init_parameters(void)
  {
    w    =  TWOPI * freq;
    em = ENORM*nl*nl*nels*nels*plswd*code_length(code)/TINC;
    x  = nels*tels;
    sf = nsrs/x;    /*  printf(" Shift-clock frequency = %f GHz",sf); */
    sp = x/nsrs;
    tmin = (1 - nels)*tels;
    tmax = plswd*code_length(code) + (nels -1)*tels;
    return;
  }


void direction_parameters(double index_direction, double azmdir)
   {
        double dir, y,z;
        dir = rd*index_direction;
        tx = tels*sin(dir);
        ty = tels*sin(rd*azmdir)*cos(dir);
        pd = w*tx;
        pl = nphs*tels*freq*sin(dir);

  /*    tdir = tels*sin(dir);
        y = nsrs;   z = nels;
        pn = ( y/z) * sin(dir);         */
        return;
    }

/* -------------------------------------------------------------------------- */


/* -------------------------------------------------------------------------- */

unsigned hash(char *s, unsigned mult, unsigned size)
 {
  unsigned hashval,i;
  for (hashval=0; *s != '\0' ; s++ )
   {
    hashval = *s + ( (mult * hashval) % size);
    i = (hashval % size);/* printf("T: %c , %d , %d , %d, %d \n",*s,i,mult,size,hashval); */
    }
   return (hashval % size);
 }


double time_a(long j) /*{ang=(long)(1000*j*ANGINC/1000.0);return(tels*sin(ang*rd));} */
    {
      long l;
      double a,da;
      l = (((1000*j)*ANGINC) + 0.0);             /* This gets rid of ugly angles... */
      ang = l/1000.0;
      a = ang*rd;
      da = tels*sin(a);
      return(da);
    }

double time_b(long j){ return(tels*sin(azmpln*rd)*cos(rd*ang));}

double time_bb(long j){ return( tels*sin(rd*ang)*cos(rd*elvpln) );}
```

```
long code_tbl( long code_index )
  {
    switch(code_index)
      {
        case 0 : return(1);          /* 1 */
        case 1 : return(2);          /* 10 */
        case 2 : return(6);          /* 110 */
        case 3 : return(114);        /* Barker-7 */
        case 4 : return(1810);       /* optimum-11 */
        case 5 : return(7989);       /* optimum-13 */
        case 6 : return(60622224);   /* UM/ONR-26 */
        default : printf(" Unknown code specified ");
      }
     return(0);
   }


void ant_file_open(char *s)
 {
   char buffer [256];
   sprintf(buffer,"%s(var)-%3.1d Data(%2.1f)",s,nels,ANGINC);
   fp = fopen(buffer,"w");
   printf("  Start of %s  Data \n", s );
   return;
 }

void study_file_open(char *file_name)
 {
   char buffer [256];
   sprintf(buffer,"Beam Pattern - %s",file_name);
   fp = fopen(buffer,"w");
   return;
 }

void base_header(void)
 {
   printf("------------------------------------------------------------------------\n");
   printf("    Angle      Energy     Decibels\n");
   printf("------------------------------------------------------------------------\n");
   return;
 }

/* ------------------------------------------------------------------------------------ */


/* ------------------------------------------------------------------------------------ */


 void dbprint(double e)
 {  double db;
    db = 10*log10(e);               /* Decibels */
    if(db < DBLIM) db = DBLIM;
    printf("%f, %f, %f\n",ang,e,db);
    fprintf(fp,"%f, %f, %f\n",ang,e,db);
    return ;
  }
```

```
 void inc_print(double e)
 {  double db;
    db = 10*log10(e);                /* Decibels */
    if(db < DBLIM) db = DBLIM;
    printf(",  %5.2f",db);
    fprintf(fp,",  %5.2f",db);
    return ;
  }


void header_print(double dirinc)
    {
        double idir;
        printf("----------------------------------------------------------------------\n");
        printf("                    Antenna Response -- Decibels\n");
        printf("----------------------------------------------------------------------\n");
        printf("Obsrv.                    Steered Angle (degrees)\n");
        printf("Angle");
        for (idir = 0.0; idir <= MAXDIR; idir = idir + dirinc)
          { printf("    %3.1f", idir); }
        printf("\n");
        printf("----------------------------------------------------------------------\n");
        return;
     }

void head_print(int study)
    {
        printf("----------------------------------------------------------------------\n");
        printf("                    Antenna Response -- Decibels\n");
        printf("----------------------------------------------------------------------\n");
        printf("Obsrv."); line_print(study);
        printf("Angle");  para_print(study);  printf("\n");
        printf("----------------------------------------------------------------------\n");
        return;
     }


void  line_print(int study)
    {
        switch(study)
            {
                case 1  : printf("        Steered Elevation Angle (degrees)\n"); bre
                case 2  : printf("            Type of Antenna System\n"); break;
                case 3  : printf("          Number of Discrete Phases\n"); break;
                case 4  : printf("       Number of Shift Register Stages\n"); break
                case 5  : printf("     Number of Antenna Elements/Dimension\n"); bre
                case 6  : printf("      Time(ns)  Between Antenna Elements\n"); brea
                case 7  : printf("               Carrier Frequency (Ghz) \n"); brea
                case 8  : printf("                Pulse Width (ns)\n"); break;
                case 9  : printf("     Number of Discrete Levels for Carrier\n"); br
                case 10 : printf("      Type of Code for Pulse Compression\n"); bre
                case 11 : printf("        Steered Azimuth Angle (degrees)\n"); brea
                case 12 : printf("     Amount of Raised Cosine Antenna Taper\n"); br
                default:  printf("Unknown study = %d in line_print\n",study);
            }
        return;
    }

void para_print(int study)
    {
```

```
        long i;
        double x;
        char *ant[] = { "TCW","DCW","TTD","PHA","DPA","DBS","DDD" };
        for (i = 0; i <= MXCASES; i++)
          {
            switch(study)
              {
                case 1 :  printf("     %3.1f",i *dirinc);         break;  /* DIR      */
                case 2 :  printf("      %s",ant[i]);              break;  /* ANT      */
                case 3 :  printf("       %d", 2 << i);            break;  /* NPHS     */
                case 4 :  printf("       %d",(long)((i+1)*in));   break;  /* SRSZ     */
                case 5 :  printf("       %d",(long)((i+1)*nn));   break;  /* NELS     */
                case 6 :  printf("     %4.2f",(i+1)*nt);          break;  /* TELS     */
                case 7 :  printf("     %4.2f",(i+1)*nf);          break;  /* FREQ    */
                case 8 :  printf("     %4.2f",(i+1)*np);          break;  /* PLSWD    */
                case 9 :  printf("       %d",2*i+1);                  break;  /* NLEVELS *
                case 10 : printf("       %d",code_tbl(i));            break;  /* CODE    *
                case 11 : printf("     %3.1f",i*dirinc);          break;  /* AZMDIR  */
                case 12 : printf("      %3.1f",((double)i)/MXCASES);break;    /* TAPER   *
                default:  printf(" Unknown type of study specified in studies.\n");
              }
          }
        return;
      }

void para_write(int study)
  {
   long i;
   double x;
   char *ant[] = { "TCW","DCW","TTD","PHA","DPA","DBS","DDD" };

   fprintf(fp,"Angle");
   for (i = 0; i <= MXCASES; i++)
    {
      switch(study)
       {
         case 1 :  fprintf(fp,"     %3.1f",i *dirinc);        break;  /* DIR      */
         case 2 :  fprintf(fp,"      %s",ant[i]);             break;  /* ANT      */
         case 3 :  fprintf(fp,"       %d", 2 << i);           break;  /* NPHS     */
         case 4 :  fprintf(fp,"       %d",(long)((i+1)*in)); break;  /* SRSZ     */
         case 5 :  fprintf(fp,"       %d",(long)((i+1)*nn)); break;  /* NELS     */
         case 6 :  fprintf(fp,"     %4.2f",(i+1)*nt);         break;  /* TELS     */
         case 7 :  fprintf(fp,"     %4.2f",(i+1)*nf);         break;  /* FREQ     */
         case 8 :  fprintf(fp,"     %4.2f",(i+1)*np);         break;  /* PLSWD    */
         case 9 :  fprintf(fp,"        %d",2*i+1);            break;  /* NLEVELS */
         case 10 : fprintf(fp,"       %d",code_tbl(i));       break;  /* CODE     */
         case 11 : fprintf(fp,"     %3.1f",i*dirinc);         break;  /* AZMDIR   */
         case 12 : fprintf(fp,"      %3.1f",((double)i)/MXCASES);break;/* TAPER   */
         default : fprintf(fp," Unknown type of study specified in studies.\n");
       }
     }
    fprintf(fp,"\n");
    return;
  }


void note_print(void)
{
 printf(" Parameters     \n -----------\nSTUDY  %s  \nMXCASES   %d\n",  STUDY,     MXCASES);
 printf("ANTENNA    %s   \nNPHS  %d  \nSRSZ  %d  \nNELS  %d\n",    ANTENNA,NPHS, SRSZ ,NELS);
 printf("TELS   %3.1f    \nFREQ  %3.1f    \nPLSWD %3.1f    \n",         TELS,    FREQ,   PLSWD);
```

```
 printf("NLEVELS    %d   \nCODE  %d   \nDIR          %3.1f    \n",          NLEVELS,CODE,   DIR);
 printf("AZMDIR %3.1f    \nMAXDIR      %3.1f    \nAZMPLN      %3.1f     \n",AZMDIR, MAXDIR,AZMPLN);
 printf("ELVPLN %3.1f    \nNUMDIM      %d   \nANGINC      %3.1f     \n",    ELVPLN, NUMDIM,ANGINC);
 printf("TAPER  %3.1f    \nDBLIM %3.1f     \n",     TAPER, DBLIM);
 return;
}


void note_write(void)
{
 fprintf(fp,"\n------------------\n\n");
 fprintf(fp," Parameters      \n -----------\nSTUDY      %s   \nMXCASES     %d\n",  STUDY,      MXCASE
 fprintf(fp,"ANTENNA     %s    \nNPHS   %d   \nSRSZ   %d   \nNELS    %d\n",   ANTENNA,NPHS, SRSZ ,NEI
 fprintf(fp,"TELS     %3.1f    \nFREQ   %3.1f     \nPLSWD %3.1f     \n",          TELS,    FREQ,   PLSW
 fprintf(fp,"NLEVELS      %d    \nCODE   %d   \nDIR          %3.1f     \n",          NLEVELS,CODE,   DIR)
 fprintf(fp,"AZMDIR %3.1f    \nMAXDIR      %3.1f     \nAZMPLN       %3.1f    \n",AZMDIR, MAXDIR,AZMPI
 fprintf(fp,"ELVPLN %3.1f    \nNUMDIM      %d   \nANGINC       %3.1f     \n",     ELVPLN, NUMDIM,ANGIN
 fprintf(fp,"TAPER  %3.1f    \nDBLIM %3.1f     \n",     TAPER, DBLIM);
 fprintf(fp,"\n------------------\n");
 return;
}


void study_parm_write(char *file_name)
 {
   char buffer [256];
   sprintf(buffer,"Beam Legend - %s",file_name);
   fp = fopen(buffer,"w");
   fprintf(fp,"Beam Legend - %s\n\n",file_name);
   note_write();
   fclose(fp);
   printf("Beam Legend: %s\n",file_name);
   return;
 }

/* ------------------------------------------------------------------------------------ */
```

DISTRIBUTION LIST

Director of Space and SDI Programs
SAF/AQSC
1060 Air Force Pentagon
Washington, DC 20330-1060

CMDR & Program Executive Officer
U S Army/CSSD-ZA
Strategic Defense Command
PO Box 15280
Arlington, VA 22215-0150

Superintendent
Code 1424
Attn Documents Librarian
Naval Postgraduate School
Monterey, CA93943

Director
Technology Directorate
Office of Naval Research
Room 407
800 N. Quincy Street
Arlington, VA 20305-1000

DTIC [2]
8725 John Jay Kingman Road
Suite 0944
Fort Belvoir,VA 22060-6218

Dr Albert Brandenstein
Chief Scientist
Office of Nat'l Drug Control Policy
Executive Office of the President
Washington, DC 20500

Dr H Lee Buchanan, I I I
Director
DARPA/DSO
3701 North Fairfax Drive
Arlington, VA 22203-1714

Dr Collier
Chief Scientist
U S Army Strategic Defense Command
PO Box 15280
Arlington, VA 22215-0280

D A R P A Library
3701 North Fairfax Drive
Arlington, VA 22209-2308

Dr Victor Demarines, Jr.
President and Chief Exec Officer
The MITRE Corporation
A210
202 Burlington Road
Bedford,M A 01730-1420

Mr Dan Flynn [5]
OSWR
Washington, DC 20505

Dr Paris Genalis
Deputy Director
OUSD(A&T)/S&TS/NW
The Pentagon, Room 3D1048
Washington, DC 20301

Dr Lawrence K. Gershwin
NIC/NIO/S&T
7E47, OHB
Washington, DC 20505

Dr Robert G Henderson
Director
JASON Program Office
The MITRE Corporation
1820 Dolley Madison Blvd
Mailstop W553
McLean, VA 22102

DISTRIBUTION LIST

Dr William E Howard I I I [2]
Director of Advanced Concepts &
Systems Design
The Pentagon Room 3E480
Washington, DC 20301-0103

J A S O N Library [5]
The MITRE Corporation
Mail Stop W002
1820 Dolley Madison Blvd
McLean, VA 22102

Dr Anita Jones
Department of Defense
DOD, DDR&E
The Pentagon, Room 3E1014
Washington, DC 20301

Mr. O' Dean P. Judd
Los Alamos National Laboratory
Mailstop F650
Los Alamos, NM 87545

Dr Bobby R Junker
Office of Naval Research
Code 111
800 North Quincy Street
Arlington, VA 22217

Dr Ken Kress
Office of Research and Development
809 Ames Building
Washington, DC 20505

Lt Gen, Howard W. Leaf, (Retired)
Director, Test and Evaluation
HQ USAF/TE
1650 Air Force Pentagon
Washington, DC 20330-1650

Mr. Larry Lynn
Director
DARPA/DIRO
3701 North Fairfax Drive
Arlington, VA 22203-1714

Dr. John Lyons
Director of Corporate Laboratory
US Army Laboratory Command
2800 Powder Mill Road
Adelphi,MD 20783-1145

Col Ed Mahen
DARPA/DIRO
3701 North Fairfax Drive
Arlington, VA 22203-1714

Dr. Arthur Manfredi
OSWR
Washington, DC 20505

Mr James J Mattice
Deputy Asst Secretary
(Research & Engineering)
SAF/AQ
Pentagon, Room 4D-977
Washington, DC 20330-1000

Dr George Mayer
Office of Director of Defense
Reserach and Engineering
Pentagon, Room 3D375
Washington, DC 20301-3030

Dr Bill Murphy
ORD
Washington, DC 20505

DISTRIBUTION LIST

Dr Julian C Nall
Institute for Defense Analyses
1801 North Beauregard Street
Alexandria, VA 22311

Dr Ari Patrinos
Director
Environmental Sciences Division
ER74/GTN
US Department of Energy
Washington, DC 20585

Dr Bruce Pierce
USD(A)D S
The Pentagon, Room 3D136
Washington, DC 20301-3090

Mr John Rausch [2]
Division Head 06 Department
NAVOPINTCEN
4301 Suitland Road
Washington, DC 20390

Records Resource
The MITRE Corporation
Mailstop W115
1820 Dolley Madison Blvd
McLean,VA 22102

Dr Victor H Reis
US Department of Energy
DP-1, Room 4A019
1000 Independence Ave, SW
Washington, DC 20585

Dr Fred E Saalfeld
Director
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

Dr Dan Schuresko
O/DDS&T
Washington, DC 20505

Dr John Schuster
Technical Director of Submarine
and SSBN Security Program
Department of the Navy OP-02T
The Pentagon Room 4D534
Washington, DC 20350-2000

Dr Michael A Stroscio
US Army Research Office
P. O. Box 12211
Research Triangle NC27709-2211

Ambassador James Sweeney
Chief Science Advisor
USACDA
320 21st Street NW
Washington, DC 20451

Dr George W Ullrich [3]
Deputy Director
Defense Nuclear Agency
6801 Telegraph Road
Alexandria, VA 22310

Dr. David Whelan
Director
DARPA/TTO
3701 North Fairfax Drive
Arlington, VA 22203-1714

Dr Edward C Whitman
Dep Assistant Secretary of the Navy
C3I Electronic Warfare & Space
Department of the Navy
The Pentagon 4D745
Washington, DC 20350-5000